Discovering and Recognizing Student Interaction Patterns in Exploratory Learning Environments

Andrea Bernardini[1] , Cristina Conati[2]

[1]Fondazione Ugo Bordoni, Via B. Castiglione 5900142, Roma, Italy.
bernardini.andrea@gmail.com

[2]University of British Columbia, 2366 Main Mall, Vancouver, BC, V6T1Z4, Canada
conati@cs.ubc.ca

Abstract. In a Exploratory Learning Environment users acquire knowledge while freely experiencing the environment. In this setting, it is often hard to identify actions or behaviors as correct or faulty, making it hard to provide adaptive support to students who do not learn well with these environments. In this paper we discuss an approach that uses Class Association Rule mining and a Class Association Rule Classifier to identify relevant interaction patterns and build student models for online classification.

We apply the approach to generate a student model for an ELE for AI algorithms and present preliminary results on its effectiveness

Keywords: Educational Data Mining, Student Modeling, Exploratory Learning Environments

1 Introduction

Exploratory learning environments (ELEs) provide functionalities such as interactive simulations and visualizations for student-led exploration of a target domain. The idea is to promote active discovery of knowledge, which in turns triggers deeper understanding of the target domain than more controlled instruction. Research however, has suggested that the pedagogical effectiveness of an ELE is highly dependent on the student who uses it: while some students appreciate the independence afforded by of this learning activity, others suffer from the lack of structure and would benefit from more guidance during interaction [1]. Such findings highlight the need for ELEs to provide adaptive support for students with diverse abilities or learning styles.

One of the challenges of providing this support is the difficulty in identifying student behaviors that warrant interventions vs. behaviors that indicate an effective learner. Traditional approaches based on creating datasets of human-labeled patterns [2][3] that can be used to classify new users are often unfeasible, because they need a priori definitions of relevant behaviors when there is limited knowledge of what these behaviors may be.

In this paper, we explore an alternative approach that relies on mining Class Association Rules to automatically identify common interaction behaviors and then uses these rules to build a user model based on a Class Association Rule Classifier. In previous work, we presented a version of the approach that used clustering algorithms to first identify learner types based on the potential effectiveness of their interaction behaviors, and then to classify new students in real time based on these clusters [11]. While the approach showed good classification accuracy in terms of the student's learning outcomes, it does not allow the system to isolate the specific behaviors that cause a given student to learn effectively or not from the environment. The approach based on association rules and a class association rule classifier that we present in this paper has a finer classification granularity, and thus it is better suited at guiding adaptive interventions to improve interaction effectiveness. We test the approach on AISpace[4], an ELE that uses interactive visualizations to help students understand Artificial Intelligence (AI) algorithms.

Several other researchers have looked at using association rules in ITS. To our knowledge, however, ours is the first attempt at using this approach for on-line student modeling with an ELE. In [5], logged data from students interaction with an ITS for the SQL database were mined using association rules to discover error patterns that can help teachers improve their presentation of this topics. In [6], the authors use association rules to discover similarities among exercises in terms of solution difficulty by mining logs of student solutions in an ITS. [7] uses association rules for the off-line analysis of students usage of a web based educational system spanning a complete university course, once the course is complete. Further off-line processing of the rules generates recommendations for teachers as to which usage patterns are more relevant for course revision.

The paper is structure has follows. We first describe our general approach to detect and recognize relevant interaction patterns in ELEs. We then introduce the ELE we used in this research. Next, we describe association rules and how they are used in our approach, and we presents results on their effectiveness in identifying effective and ineffective learners in AISpace. We conclude with a discussion of future work.

.

2 General Student Modeling Approach

Our student modeling approach for ELEs divides the modeling process into two major phases: offline identification and online recognition.

In the offline phase, raw, unlabelled data from student interaction with the target environment is first collected and then preprocessed. The result of preprocessing is a set of feature vectors representing individual students in terms of their interaction behavior. These vectors are then used as input to an unsupervised clustering algorithm that groups them according to their similarity. The resulting groups, or 'clusters', represent students who interact similarly with the environment. These clusters are then analyzed to determine which interaction behaviors are effective or ineffective for learning. The analysis consists of first identifying how the different clusters relate to learning outcomes, and then isolating in each cluster those behaviors that are responsible for the learning effects. Understanding the effectiveness of students' interaction behaviors with an ELE is useful in itself to increase educator awareness of the pedagogical benefits of these environments, as well as to reveal to developers how the ELE can be improved [8][9][10]. However, our long-term goal is to use the interaction behaviors to guide automatic ELE adaptations while a student is interacting with the system. Thus, in the online recognition phase, the clusters identified in the offline phase are used directly in a classifier user model. The user model's classifications and the learning behaviors identified by cluster analysis can eventually be used to inform an adaptive ELE component to encourage effective learning behaviors and prevent detrimental ones.

In our previous investigations of this approach [11], the identification of behaviors that influence learning outcomes in each cluster was done by hand, using formal statistical tests to evaluate cluster similarities and dissimilarities along each of the feature dimensions. The outcome of this step is useful to help educators and developers gain insights on the different learning behaviors and design appropriate adaptive interventions targeting them. It was not, however, directly used during on-line learner recognition. For this phase, we devised an online *k*-means classifier, trained on the clusters identified in the offline phase. This classifier incrementally updates the classification of a new student into one of the clusters from the offline phase, as the student interacts with the target ELE. While this classifier showed very good performance in predicting student learning outcomes with two different ELEs, it can't identify which behaviors caused the classification at any given time. For instance, when applied to the ELE that we describe later in this paper, this approach identifies two clusters, one of high learners and one of low learners. Each cluster includes a variety of behaviors that can impact learning but the classifier can't tell which behaviors are responsible for the student's classification as student actions come in. This limits the ability of the approach to support adaptive interventions that target the relevant behaviors (e.g. discourage superficial browsing of functionalities). To address this limitation, we have introduced the use of Class Association Rules, described in the next section.

3 Using Class Association Rules for Learner Classification in ELEs

Association rules were originally devised for finding the hidden connections between items in a transaction database[12], and are generally used to find co-occurrence patterns in data. The connections are expressed as rules *X --> Y*, indicating that when *X* occurs, *Y* occurs with a given probability greater than zero. This probability is called the *confidence* of the rule *conf(R)*, which essentially represents the strength of the correlation between X and Y. Algorithms for generating association rules use this measure, along with a measure of the relevance of a rule in a dataset, to select a set of appropriate rules among a usually very large pool of candidates. The measure of relevance is commonly known as *support* of the rule *sup(R)*, computed as the percentage of datapoints satisfying both X and Y in the dataset.

The use of association rules to construct a classifier is called Associative classification mining or Associative Classification [13]. Algorithms for Associative Classification usually operate by first generating a complete set of class association rules (CARs) from training data, and then by pruning this initial set to obtain a subset of rules that constitute the classifier. When a new unknown object (a student in our case) is presented to the classifier, it is compared to a number of CARs and its class is predicted. The selection of the representative subset of CARs is one of the crucial steps in Associative Classification, entailing understanding what is the best number of CARs needed for classification, as well as which measures to use to select them.

We use Associative classification to increase the grain size of the on-line classification in the student modeling approach we described in Section 2. The overall approach is modified as follows. We still rely on an off-line clustering algorithm for generating the clusters that form the basis for the classifier, and each cluster is labeled with the overall learning performance of the corresponding student group. The off-line phase is then augmented by performing Associative Classification within each cluster, thus obtaining a set of CARs that, for each cluster, link specific interaction behaviors with learning outcomes. During on-line classification, user interaction behaviors are tracked and updated after each action, as before. This time, however, they are matched against all available CARs and a classification is selected based on the cluster that has the highest percentage of matched rules. Thus, at any given point of the interaction, our student models generates a prediction of both the current student's learning success, as well as the behaviors that influence it.

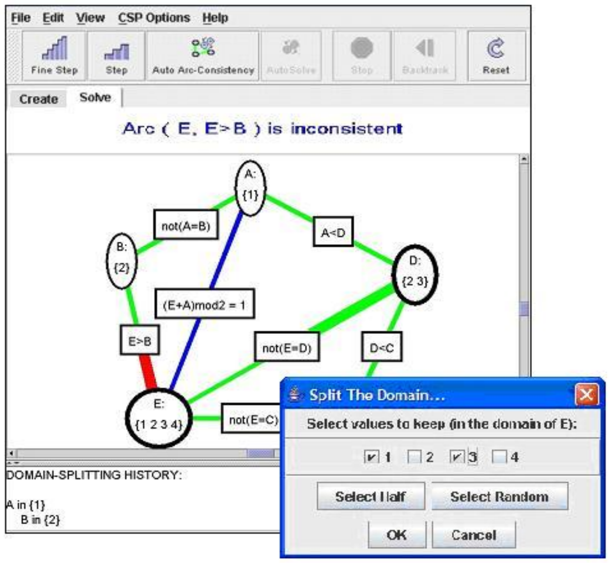
In the rest of the paper, we provide details on the approach and its effectiveness in the context of its usage for modeling students as they interact with the ELE known as AISpace CSP applet, described in the next section.

4 The AISpace CSP applet

The ELE we use as a testbed for our approach is the Constraint Satisfaction Problem (CSP) Applet, one of a collection of interactive tools for learning common Artificial Intelligence algorithms, called AIspace [4]. Algorithm dynamics are demonstrated via interactive visualizations on graphs by the use of color and highlighting, and graphical state changes are reinforced through textual messages (see Figure 1 for an example).

A CSP consists of a set of variables, variable domains and a set of constraints on legal variable-value assignments. The goal is to find an assignment that satisfies all constraints. The CSP applet illustrates the Arc Consistency 3 (AC-3) algorithm for solving CSPs represented as networks of variable nodes and constraint arcs. AC-3 iteratively makes individual arcs consistent by removing variable domain values inconsistent with a given constraint until all arcs have been considered and the network is consistent. Then, if there remains a variable with more than one domain value, a procedure called domain splitting can be applied to that variable to split the CSP into disjoint cases so that AC-3 can recursively solve each case or sub-network.

The CSP applet provides several mechanisms for interactive execution of the AC-3 algorithm, accessible through the toolbar shown at the top of Figure 1 or through direct manipulation of graph elements. Here we provide a brief description of these mechanisms necessary to understand the results of applying our student modeling approach to this environment:



Arc (E, E>B) is inconsistent

Select values to keep (in the domain of E):

DOMAIN-SPLITTING HISTORY:

A in {1}
B in {2}

Fig.1 CSP applet with example CSP

- *Fine Stepping.* Cycles through three detailed algorithm steps: selecting an arc, testing it for consistency, and removing variable domain values when necessary.
- *Direct Arc Clicking.* Allows the user to decide which arc to test, and then performs three *Fine Steps* on that arc to make it consistent.
- *Auto Arc Consistency (Auto AC).* Automatically *Fine Steps* through the network.
- *Stop*. Stops *Auto AC*.
- *Domain Splitting (DS)*. Allows the user to select a variable domain to split, and specify a sub-network for further application of AC-3.
- *Backtracking*. Recovers the alternative sub-network set aside by *DS*.
- *Resetting.* Resets the CSP network to its initial state.

Currently, AI space does not provide any explicit support on how to use the available mechanisms to learn at best from the interactive visualizations delivered by its applets. Research, however, shows that students may benefit from this support, since unaided exploration of interactive visualizations often fails to help students learn[11]. In the following sections, we describe the application of the modeling approach described in Section 3 to create a classifier user model that can detect suboptimal student interactions with the CSP applet and guide adaptive interventions aimed at improving them.

5 Modeling student interaction with the CSP applet

The data we use for this research was obtained from a previous experiment investigating the effects of studying sample problems with the CSP applet. We use the following data collected from 24 students who participated in the study: time-stamped logs of user interactions with the applet, and learning gains computed from pre and post tests administered to the study participants. From the logged data we obtained 1931 actions of users over 205.3 minutes. In the off-line phase, in order to find clusters of students who interact with the CSP Applet in similar ways, each student must be represented by a multidimensional data point or ‘feature vector’. From the logged user study data, we computed 24 feature vectors corresponding to the 24 study participants. The feature vectors had 21 dimensions, resulting from deriving three features for each of the seven actions described in the previous section: (1) action frequency, (2) the average latency after an action (reported as *avg* in tables), and (3) the standard deviation of the latency after an action (reported as *STD* in tables). The latency dimensions are intended to measure if and how a student is reflecting on action results. Specifically, the second dimension is an indicator of student reflection, and the third dimension is an indicator of reflection selectiveness since varied latency may indicate planned rather than impulsive or inattentive behavior (e.g., consistently rushing through actions vs. selectively attending to the results of actions).

After forming the feature vector representation of the data, the next step in the offline phase is to perform clustering on the feature vectors to discover patterns in the students' interaction behaviors. After experimenting with various clustering algorithms (including EM and hierarchical clustering) we chose *k*-means[14] for this dataset. *K*-means converges to different local optima depending on the selection of the initial cluster centroids and so in this research we execute 25 trials (with randomly selected initial cluster centroids) and use the highest quality clusters (based on Fisher's criterion [15]) as the final cluster set. We also experimented with *k* set to 2, 3 and 4, and obtained the best results for $k = 2$. More details on cluster generation can be found in [11].

When we compared average learning gains between the two clusters found by *k*-means, we found that one cluster (4 students) had statistically significantly higher learning gains (7 points) than the other cluster (20 students, 3.08 points gain). Hereafter, we will refer to these clusters as ‘HL’ (high learning) cluster, and ‘LL’ (low learning) cluster respectively.

5.1 CARs and Multiple Class Association Rule Classifier

The next step in the student modeling process is to generate CARs to identify, for each cluster, the interaction behaviors that best characterize its students. In this work, we used the Hotspot algorithm in Weka [16], which inspects the training data and generates the association rules corresponding to a class label in the form of a tree. Table 1 shows the CARs generated for the HL and LL clusters, where we report the preconditions for each rule but leave out the consequence (Label HL for the high learners cluster and LL for the low learners cluster). Table 1 also shows, for each rule, its level of confidence (*conf*), and support within its cluster (*supp*).

It should be noted that the attribute values mentioned in the rules in table 1 are discrete, while our original dimensions are continuous. Although CAR algorithms work with both discrete and continuous values, using continuous values in our dataset would produce a large number of very fine-grained rules, unsuitable for classification. We thus discretized our attributes using the equal-width method proposed in [17], which consists of dividing the range of observed values into k equally bins. For the time being we selected k=2, thus discretizing each attribute into HIGH and LOW values, although we plan to experiment with a higher number of bins to see how that affects the accuracy of the classifier.

Table 1 CARs for HL and LL clusters (*STD* refers to standard deviation, *Avg* referes to average)

Rules for HL cluster

Rule 1*: Stop Pause STD = HIGH; conf=100%; supp =50%*
Rule 2*: Fine Step Pause STD = HIGH; conf=80% supp =100%*
|***Rule 3*** *Fine Step Pause STD = HIGH & Fine Step frequency = LOW*
conf=100% supp=100%
|***Rule 4*** *Fine Step Pause STD = HIGH & Domain Split Pause STD = LOW*
conf=100% supp =100%

Rules for LL cluster

Rule 1 *Fine Step Pause STD = LOW; conf=100%; supp =100%*
Rule 2 *Stop frequency = LOW; conf= 95% ; supp =95%*
| ***Rule 3*** *Stop frequency = LOW & Fine Step Pause Avg = 'LOW*
conf=100% supp=100%
| ***Rule 4*** *Stop frequency = LOW & Reset Pause Avg = 'LOW*
conf=100% supp=100%

The Hotspot algorithm has three parameters that influence the type and number of rules generated: the minimum level of support requested for a rule to be considered relevant; the branching factor of the tree, influencing how many new rules can be generated from an existing one by adding a new condition to its current set; the minimum improvement in confidence requested for creating a new branch in the tree.

We kept the default values for minimum improvement (0.01) and branching factor (2), while we used the minimum level of support within each cluster as a criterion to filter the number of rules generated [18]. Essentially, for each cluster we need to find a few rules that characterize as many elements in the cluster as possible and provide an easily understandable explanation of students' behaviors for each learning outcome. After experimenting with various levels of support, we selected 50% for both the HL and the LL cluster, i.e., a rule has to involve at least half of the students in the cluster to be generated.

The CARs produced are shown in Table 1. They indicate that, for instance, high learners show more selective attention when observing the workings on the CSP algorithm in the applet (see high values for standard deviation in latency after stopping a running of Autosolve in Rule 1, and in latency during fine stepping in Rules 2-4). Low learners, on the other hand, are characterized by non-selective attention during fine stepping (see low standard deviation for pausing times during stepping in rule 1). Rule 2 represents a different detrimental behavior, i.e. short latency during stepping and limited usage of stopping during autosolving, indicating that the student is not taking the time to analyze the workings of the algorithm.

After producing the CARs for each cluster, the CARs were used as input to a classifier based on multiple class association rules. This classifier can generate a prediction in terms of high or low learning for a new user after each user action in the CSP applet. For each new action, all the related feature dimensions are recomputed (e.g., action frequency and the various latency dimensions), and the updated feature vector is matched against all existing CARs. The final classification is generated by selecting the cluster with the highest percentage of matched rules.

5.2 Evaluation

To evaluate our approach, we used the same methodology followed in [11] and based on a 24-fold leave-one-out cross validation (LOOCV). In each fold, we removed one student's data from the set of *N* available feature vectors, used *k*-means to re-cluster the reduced feature vector set and generated CARs for each newly generated cluster. Next, the removed student's data (the test data) was fed into the CAR Classifier trained on the reduced set, and online predictions were made for the incoming actions as described above. Model accuracy is evaluated by checking after every action whether the current student is correctly classified into the cluster to which he/she was assigned in the offline phase. The percentage of correct classifications is shown in Figure 2 as a function of the percentage of student actions seen by the model (solid line labeled 'Overall' in the figure's legend). The figure also shows the model's performance in classifying HL students into the HL cluster (dotted line), LL students into the LL cluster (dashed line), and the performance of a base-line most-likely classifier.

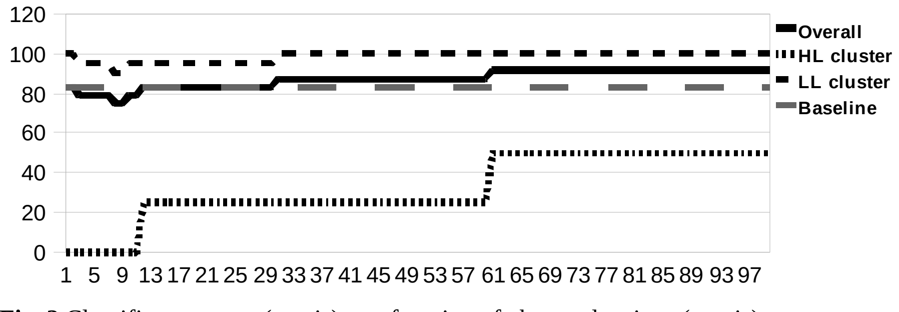


Fig. 2 Classifier accuracy (y axis) as a function of observed actions (x-axis)

The classifier performs very well in identifying LL learners (100% accuracy after seeing about 30 actions), while it performs poorly with the HL learners. This is not surprising, since the HL cluster used to derive the CARs rule for this group contains on average only 3 data points during LOOCV. The high performance on the LL cluster, on the other hand, is very encouraging, because this cluster, although much larger than HL, still includes a relatively limited number of datapoints (20 on average). Still, we managed to learn from this dataset a CAR classifier that is very good at detecting students with suboptimal learning behaviors, indicating that overall performance can be significantly improved by collecting a richer dataset for model training.

It is interesting to compare the performance of our CAR Classifier with the performance of the previous k-means classifier [11]. That classifier had slightly lower accuracy on LL (constantly above 90% but never reaching100%) but scored much better than the CAR classifier in classifying HL, converging to about 75% accuracy after seeing about 40% of the available actions. These results suggests that the k-mean classifier learns more reliably from small datasets that the CAR classifier, but the price to pay is no information on which behaviors are responsible for a user's classification at any given point of the interaction. The CAR classifier, on the other hand, can help identify these behaviors after every user action by providing the rules that were matched to generate the classification. This information can be used to provide adaptive hints to correct behaviors that can be detrimental for learning. For instance, if Rule 3 below fires to classify a student as low learner (see Table 1)

Rule 3 *Stop frequency = LOW and Fine Step Pause Avg = 'LOW*

the ELE can try to make the learner stop more often when running the CSP algorithm in autosolve mode, and pause more in between stepping actions to reflect on the outcome of each step of the algorithm

6 Conclusions and Future Work

We presented a student modeling approach that uses Class Association Rules and a Class Association Rule Classifier to discover and monitor student behaviors that can impact learning during interaction with an ELE. Modeling student interactions with ELEs is important to provide adaptive support for those students who do not learn well in absence of more structured instruction. It is also challenging, because in these environments it can be hard to identify a priory actions or behaviors as correct or faulty. We have provided initial results showing that our approach can identify these behaviors, and have discussed implications for providing adaptive support in ELE. One line of future work, thus, is to implement this adaptive support. In parallel, we want to refine our student modeling approach by experimenting with alternative techniques for selecting the set of relevant association rules (e.g. based on a variety of functions of support and confidence). We also want to see how our approach performs on larger datasets and how it transfers to different ELEs, for instance the ELE for mathematical functions that we used to test transfer of the first version of our approach without association rules [11].

References

1. Shute, V. J. : A comparison of learning environments: All that glitters... In: Computers as Cognitive Tools. pp. 47--73 Lawrence Erlbaum Associates, Hillsdale (1993)
2. Merten, C., Conati, C.: Eye-Tracking to Model and Adapt to User Meta-Cognition in Intelligent Learning Environments. In Proceedings of Intelligent User Interfaces (2006)
3. Baker, et al.: Adapting to When Students Game an Intelligent Tutoring System. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems. pp. 392-401(2006)
4. Amershi S., Carenini G., Conati C., Mackworth A., Poole D. . Pedagogy and Usability in Interactive Algorithm Visualisations: Designing and Evaluating CIspace. *Interacting with Computers*, 20(1), 64-96 (2008)
5. Merceron A., and Yacef K.: A web-based tutoring tool with mining facilities to Improve Teaching and Learning, AIED 2003, 201--208 (2003)
6. Freyberger, J., Heffernan, N., Ruiz, C.: Using association rules to guide a search for best fitting transfer models of student learning. Workshop on analyzing student-tutor interactions logs to improve educational outcomes at ITS conference, 1--4. (2004)
7. García, E., Romero, C., Ventura, S., and Castro, C. D.: An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. UMUAI 19, 99--132 (2009)
8. Hunt, E., Madhyastha, T.: Data Mining Patterns of Thought. In Proceedings of the AAAI Workshop on Educational Data Mining.(2005)
9. Merceron, A., Yacef, K.: TADA-Ed for Educational Data Mining. Interactive Mulitmedia Electronic Journal of Computer-Enhanced Learning (2005)
10. Talavare, L., Gaudioso, E.: Mining Student Data to Characterize Similar Behavior Groups in Unstructured Collaboration Spaces. In Proceedings of the European Conference on AI Workshop on AI in CSCL.(2004)
11. Amershi, S. and Conati, C.: Combining Unsupervised and Supervised Machine Learning to Build User Models for Exploratory Learning Environments. In: The Journal of Educational Data Mining, 1(1), 18-71 (2009).
12. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in large Databases,VLDB 1994, pp. 487--499(1994)
13. Thabtah, F.A.: A review of associative classification mining. Knowl. Eng. Rev. 22, 1, pp 37--65 (2007)
14. Duda, R. O., Hart, P. E., Stork, D., G.: Pattern Classification. New York: Wiley-Interscience (2001)
15. Fisher, R. A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7, pp. 179--188 (1936)
16. Weka project, http://www.cs.waikato.ac.nz/~ml/weka/
17. Dougherty, J., Kohavi, R., and Sahami, M.: Supervised and unsupervised discretization of continuous features. In Proceedings of the 12th International Conference on Machine Learning, pp. 194--202(1995)
18. Liu, B., Hsu, W., and Ma, Y.: Mining association rules with multiple minimum supports. In Proceedings of the Fifth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining. KDD '99. ACM, pp. 337--341(1999)