A CLUSTER-BASED PREDICTIVE MODELING TO IMPROVE PEDAGOGIC REASONING

Roberto Legaspi  
The Institute of Scientific and Industrial Research, Osaka University  
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan

Raymund Sison  
College of Computer Studies, De La Salle University-Manila  
2401 Taft Avenue, Manila, 1004, Philippines

Ken-ichi Fukui  
The Institute of Scientific and Industrial Research, Osaka University  
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan

Masayuki Numao  
The Institute of Scientific and Industrial Research, Osaka University  
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan

ABSTRACT
This paper discusses a cluster knowledge-based predictive modeling framework actualized in a learning agent that leverages on the capability of a clustering algorithm to discover in logged tutorial interactions unknown structures that may exhibit predictive characteristics. The learned cluster models are described along learner-system interaction attributes, i.e., in terms of the learner’s knowledge state and behaviour and system’s tutoring actions. The agent utilizes the knowledge of its various clusters to learn predictive models of high-level student information that can be utilized to support fine-grained individualized adaptation. We investigated on utilizing the Self-Organizing Map as clustering algorithm, and the naïve Bayesian classifier and perceptron as weighting algorithms to learn the predictive models. Though the agent faced the difficulty imposed by the experimentation dataset, empirical results show that utilizing cluster knowledge has the potential to improve coarse-grained prediction for a more informed and improved pedagogic decision-making.

KEYWORDS
Educational data mining, Machine learning in Intelligent Tutoring Systems, Learner modeling.

1. INTRODUCTION AND RESEARCH GOAL
Principled methodologies for developing teaching expertise in computerized teaching systems include the implementation of a learning theory and the observation of human teachers followed by an encoding of effective examples of these teachers’ expertise (du Boulay & Luckin, 2001). Unfortunately, these pedagogical and/or cognitive theories and examples with human instructors in mind may not at all be optimal for use with computers. They may need to be further studied and evaluated extensively within the context of educational software (Beck et al, 2000). User testing may reveal that students are not at all learning as predicted and that the teaching decisions that adopt these theories and examples are deficient. Most teaching decisions are based on production rules, and to dynamically adapt rule-based paradigms is difficult.
Consequently, system implementers will revise the rule composition and selection mechanisms off-line to achieve an improved version. One way to overcome this is to provide learning capability to the system. Research enabling teaching systems to utilize their interaction experiences with students as catalytic background knowledge to improve their teaching behaviour over time is gaining attention (e.g., Dillenbourg, 1989; Stern et al, 1999; Beck et al, 2000; Eliorraga & Fernandez-Castro, 2000; Mayo & Mitrovic, 2001; Arroyo et al, 2004). The primary concern is the point, or points as the behaviour becomes regular, in time where the system needs to analyze its prior interactions with students in order to infer new or modified ways of teaching, and then evaluate these in succeeding interactions. Systems with such learning capabilities log student and tutor actions in a large database and then use machine learning paradigms to infer useful knowledge from the data. It is only recently that this approach was given the rubric educational data mining.

We are working on a learning framework that leverages on the capability of a clustering algorithm to learn unknown structures in logged tutorial interactions, and then uses the inferred cluster knowledge for high-level prediction to support fine-grained level individualized adaptation. Clustering is an inductive method that produces a categorization scheme over a set of unlabeled data objects. This framework is actualized in MORPHUS – a polymorphous cluster knowledge-based predictive model learning agent that can support an intelligent tutoring system (ITS) to improve its pedagogic reasoning. MORPHUS achieves this in two ways:

1. It data mines history logs of fine-grained ITS-student interactions and automatically form cluster models that are characterized along the following dimensions:
   - Each cluster is described along learner-system interaction attributes, i.e., in terms of the learner’s knowledge state and behaviour and system’s tutoring responses. This gives way for a cluster-based pedagogic reasoning using attributes that are beyond those that commonly make up a student model.
   - Secondly, this allows MORPHUS to give different predictions for different instances or cases even if the learner is the same, thereby distinguishing it from the categorization approach to student modeling.
   - Any of the formed models can be polymorphous, i.e., cluster members are described in terms of similar features (attribute-value pair), but also permits values of the same attribute to differ.
2. It utilizes its cluster knowledge together with a weighting scheme to construct predictive models of high-level student responses that can provide helpful indications of which of the system’s actions can yield high learning gain. This can be beneficial to any pedagogic decision-making entity. For example, to infer certain aspects of student’s response can provide information as to which of the alternative feedback is most appropriate for the student or determine under which conditions the student will perform best.

This paper first locates our research along with existing related works. It then discusses how the experimentation dataset was acquired and the reason for adopting high-level student information as teaching goal parameters. This is followed by an overview of the learning framework of MORPHUS and its utility. The succeeding sections discuss in depth the characteristics of the clusters formed by MORPHUS and the analysis and evaluation of its performance.

2. RELATED WORKS

A plethora of research view students having different traits to respond differently to teaching methods (as reviewed by Cronbach & Snow, 1977). Moreover, evidence show that when learners are grouped or categorized based on common characteristics that make them different from others and adaptation is made distinct on the level of these categories, the effectiveness of a teaching system increases (e.g., Arroyo et al, 2000; Ainsworth et al, 2000). Adapting teaching/learning environments to the needs of learner categories for optimal learning gains requires that learners be diagnosed and categorized along certain attributes only if it makes difference in learning, and that the categorization task be efficient. Both are difficult problems.

The cost of knowledge engineering (and expert knowledge) required to analyze which traits, whether aptitude- or attitude-related, to diagnose the students is high. In testing new hypothesis on learner trait-system treatment interaction, the number of treatment elements increases exponentially with respect to the number of treatment features, and so does the number of subjects needed to test the significance of such features. This can easily lead to an intractable student model.

Commonly, category membership is based on conjunction of learner features (e.g., all members are high cognitive ability girls). This is where certain limitations arise. Hanson and Bauer (1989) stated that finding categories based on a common feature mechanism, or conjunctive categorization, generally adopts a
membership rule that requires necessary and sufficient conditions for the category members. They also stated, however, that some categories may exist as polymorphous, i.e., they tend to possess members which have features that are neither necessary nor sufficient. This means that a category may have different values for a certain attribute, and that the significant attributes in each category may vary. For example, a conjunctive category would have high cognitive ability and visual girls, while a polymorphous category may have high cognitive ability boys and girls who are more visual and less kinesthetic. The issue is how to move away from a conjunctive categorization and still acquire useful models.

The kinds of categories that may exist may be known a priori (e.g., according to gender or age bracket). However, it may also be the case that a system aims to find a category but does not know exactly what it should look like. Therefore, it must rely solely on its categorization process to reveal unknown structures with no “teacher” to guide the learning process, hence, unsupervised.

A solution to these issues is to provide the system machine learning (ML) capability in order to simplify system construction and automatically discover unknown structures in the data whose nature may be polymorphous. ML techniques have been successfully applied to computerized tutors in various ways: to infer student models (as reviewed in Sison & Shimura, 1998), to infer high-level student information (refer to Beck & Woolf, 2000), to optimize teaching responses (e.g., Beck et al, 2000; Mayo & Mitrovic, 2001), to infer unobservable learning variables from students’ help seeking behaviour (Arroyo et al, 2004), to understand how learning proceeds through simulated students (e.g., VanLehn et al, 1994; Beck et al, 2000), and to initialize the student model (e.g., Aimeur et al, 2002; Tsiriga & Virvou, 2004), among others.

The unsupervised induction approach in ML to the problem of categorization is clustering. Traditional approaches to cluster analysis (numerical taxonomy) represent the data objects to be clustered as points in a multi-dimensional metric space and adopt distance metrics to define similarity between objects. We concentrate on conceptual cluster analysis that represents objects as vectors of attribute values and typically describes cluster structures in the multi-dimensional space as mean vectors representing cluster centers from which nearest objects are assigned to its cluster. Though the end result of clustering is taxonomy, conceptual clusterers go a step further by acquiring for its clusters conceptual descriptions that provide meaningful interpretations. Though the most influential conceptual clusterers are symbolic, i.e., their representations are directly understandable, unsupervised neural learners also exist (as discussed in Sison & Shimura, 1998).

As far as our knowledge of the literature is concerned, there is paucity of works employing unsupervised learning paradigms for categorization in ITSs. Table 1 locates our work together with existing ones:

<table>
<thead>
<tr>
<th>Attributes for Category Formation</th>
<th>Student Model Initialization</th>
<th>Treatment Adaptation and Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category Formation</td>
<td></td>
<td>Category-level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Individualized</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fine-grained</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coarse-grained</td>
</tr>
<tr>
<td>Learner</td>
<td>CLARISSE</td>
<td>AgentX</td>
</tr>
<tr>
<td>Learner-system</td>
<td></td>
<td>CSPM</td>
</tr>
</tbody>
</table>

The other three adopt a conjunctive categorization based on learner attributes. The categories of both CLARISSE (Aimeur et al, 2002) and AgentX (Martin & Arroyo, 2004) depict proficiency levels based on their learners’ pretest performance. CSPM (Legaspi, et al, 2004) makes its categories distinct through plain conjunction of learners’ learning styles, cognitive abilities, scope of domain content covered, and error/s committed. Only CLARISSE adopts unsupervised category formation, albeit does not learn any categorical treatment. Both AgentX and CSPM learn their treatments in an unsupervised way, and while both will eventually customize their treatment to a category level, they differ in the granularity of treatment. While AgentX provides hints at every step of the session, CSPM reasons on a high-level of abstraction, i.e., it only decides on what could be the best sequence of teaching activities based on the category knowledge it possesses and assumes that the teaching module is responsible for executing each activity.

MORPHUS tackles unsupervised cluster formation along learner-system attributes and the resulting cluster knowledge is used to help improve treatments on the tutorial interaction level. Because of the way its modeling task is characterized and the alternatives by which its output can be utilized, the term cluster is more fitting than category. The rest of the paper discusses the core concepts of MORPHUS and the experiments we conducted to evaluate these concepts.
3. EXPERIMENTATION DATASET

In order to conduct significant experiments under the same initial conditions, a dataset collected in real setting is required. This became for us a stumbling block in making immediate progress. Apart from the very high cost to knowledge-engineer an ITS, at this time, there is no data repository for educational data mining as opposed to the availability of shared data in other research areas (e.g., web usage mining) that makes it easy for their researchers to compare techniques without the need to implement and deploy a new system. We moved around this obstacle by borrowing\(^1\) a dataset sufficient to test our hypothesis. However, this meant adopting the objectives for which the dataset has been instrumented.

The dataset originally consisted of 11,612 instances of prior tutorial interactions of the arithmetic tutor AnimalWatch (Woolf et al, 2001) with 104 students. We pre-processed it by removing instances with uncertain features. For example, the gender attribute was removed since 50% of the instances have unknown values for it, and more than 500 instances were removed for unknown levels of cognitive development. The number of interaction instances became 11,070 with 99 students. Each instance include information about the student’s knowledge state, problem and hint characteristics, student’s current efforts in answering the problem (i.e., his responses to hints, number of mistakes, and accumulated response times), and the amount of time it took the student to answer and whether it was correct. We have a total of 28 attributes to represent each instance.

What we have is an earlier (hence, not the actual) version of the dataset used by ADVISOR (Beck et al, 2000). The complete features of the actual dataset are discussed in (Beck & Woolf, 2000). ADVISOR is a two-agent machine-learning (ML) architecture that replaced the heuristics that govern the teaching strategies of AnimalWatch with ML-derived strategies. The first agent used two linear regression models – one to predict the amount of time for the student’s next response (in log-milliseconds so as to linearize the time), and the other outputs the probability that this next response would be correct. It makes a short range prediction: INPUTS: \(\text{Student} \times \text{Problem} \times \text{Hint} \times \text{Efforts} \rightarrow \text{OUTPUTS: } \text{the student will answer correctly with probability } P \text{ after time } T\). This prediction is then used by the second agent to compute for optimal teaching strategies. To determine if the first agent is sufficiently accurate, Beck et al. (2000) compared its predictions to how the students in their training dataset actually performed. They correlated the agent’s predictions with actual performance.

We have adopted the goal of the first agent as ready objective for MORPHUS. Observing time and correctness of student’s response is a relatively simple set of goals at a tractable level as opposed to a cognitive process level which cannot be observed directly (Woolf et al, 2001). To evaluate MORPHUS, we correlated its predictive time values with the actual, and instead of correlating the probability, we computed for the average of correct matches between the predictions of MORPHUS and the students actually getting the correct answer or not. The novel spirit within MORPHUS is that of inferring clusters from traces of student-ITS interactions without explicitly making sense of what characterizes each cluster, and later on, in a new way, utilizing the knowledge derived from these clustered traces.

4. LEARNING FRAMEWORK AND POSSIBLE USE

Figure 1 shows MORPHUS and two possible means by which it can provide useful cluster-based predictions.

To learn models of conceptual clusters, MORPHUS employs the Self-Organizing Map (SOM, Kohonen, 2001)\(^2\). SOM efficiently learns optimal reference vectors that act as centers of gravity for the clusters. These vectors update themselves by learning to become more like the interaction instances they each aim to represent, consequently creating clusters in a map representing similarities. Later on, an instance shall be classified to the cluster

---

\(^1\) We are grateful to Dr. Joseph Beck for providing us the dataset to test and evaluate MORPHUS.

\(^2\) We used the the Self-Organizing Map Program Package Version 3.1; URL:http://www.fdp.funit.fi/pub/sci/neural/cochlea/som_pak/
whose reference vector is least distant to its feature vector. SOM attempts to visualize highly dimensional data in a two-dimensional map. In this way, we can confine ourselves to the two-dimensional map that apparently reveals essential relationships in the data. SOM is among the model-based approaches that offer interpretability since the resulting model for each cluster directly characterizes that cluster (Zhong & Ghosh, 2003).

We investigated on two weighting algorithms, namely, the naïve Bayesian classifier and the perceptron. Despite the simplifying assumptions that underlie the naïve Bayesian classifier (Langley et al, 1992), studies have shown that it is competitive with much more sophisticated induction algorithms. It is based on probability models that incorporate strong independence assumptions and whose goal is to accurately predict the class of test instances and in which the training instances include class information. We used this classifier to predict response correctness. Once an instance is classified to an existing cluster, this classifier is then used to further classify the instance to the class of correct, or to the class of wrong, answers within the cluster, labelled as 1 and 0, respectively. Consequently, the resulting class becomes the predictive value for response correctness. However, since class labels need to be discrete for this classifier, we cannot use it to predict response time. Instead, we looked into the perceptron.

The perceptron (Rosenblatt, 1959) is a single-layer network that differentiates sets of data in a supervised way. The idea is to adjust the weights of the network to minimize some measure of error, E, on the training set. We used gradient descent to reduce the squared error by calculating the partial derivative of E with respect to the weight. The weight is iteratively modified until E is less than a user-specified threshold or a predetermined number of iterations have been completed. In the case of MORPHUS, cluster knowledge, i.e., the clusters’ feature vectors that include the desired parameter values, is submitted to the perceptron as training data. Connection weights exist, and need to be updated, for every attribute and that these weights will differ for each cluster. Consequently, a predictive parameter value for a new instance is computed as the dot product of the instance’s feature vector and the predictive weight vector corresponding to the cluster to which the instance is classified. Since we are at it, we also looked into predicting response correctness using the perceptron. Its predictive value is rounded off to 1 or 0.

The predictive models of MORPHUS can be utilized basically to estimate the effects of alternative tutoring actions as manifested in the student’s response in order to select what seems to be best. Prediction performance must be high if the next best action is to be determined on the fly. Furthermore, this capability is potentially nearsighted, i.e., it does not consider the possibility that what seems to be an optimal action now may lead to a bad or suboptimal situation later. Beck et al’s (2000) solution is to translate such capability to a student simulation for a pedagogical agent to learn proactive strategies so that the learner stays in a desirable situation. If the goal is to provide a simulation for another agent to learn improved or new strategies, or to make initial predictions and aim to increase predictive power by updating the predictive models as succeeding interaction logs are incorporated, then MORPHUS can provide the needed capability.

5. POLYMORPHOUS CHARACTER OF THE FORMED CLUSTERS

Figure 2 shows the polymorphous character that can be observed among sample clusters in one of the actual models formed by MORPHUS. The distance from the center of any point (each point is associated to a cluster) along an attribute axis in the radar-like graph indicates the standard deviation among the values for that attribute. Figure 2a summarizes the characteristics of nine actual clusters whose instances differ only in terms of learner
descriptions, and problem difficulty level, operand size, and utility. These actually exemplify the first stage of student-system interaction where AnimalWatch selects problems based on learners’ knowledge state. It also shows that a problem’s difficulty level changes as its utility and operand size are changed. In AnimalWatch, the difficulty levels for each topic are handled by adjusting the utilization of skills (e.g., borrowing in whole number subtraction) and operand size. No hint was provided since at this stage the student has yet to make his initial action. Descriptions of this kind can help select problems that would not be too hard or too easy for the student. Figure 2b shows two clusters whose learners react differently to different problems when provided similar hints. These indicate the varying effects of a hint to various learners given different problems. The four clusters in Figure 2c are more polymorphous as all features vary in certain degrees as indicated by their points. With this polymorphous character, it can be expected that learners’ high-level responses will vary.

We shall refer to the time it took the student to respond and whether it is correct (1-correct, 0-incorrect) as RT and RC, respectively. Figure 3 shows the range of RT (in log-milliseconds) and the probabilities of RC=1 for each cluster. Figure 3a shows that there are areas where RT values in a cluster are close at certain ranges, while at other points, they are dispersed. Figure 3b shows, however, that only about 12% (in the boxed region) of the clusters do not exhibit majority (mostly 1 or 0) values for RC. A pattern seems to be more likely for RC. The question here is why not just cluster the instances having the same learner-system features? We can anticipate the students’ response features per cluster to be similar and MORPHUS can simply take their averages as the predictive values. This, however, is not the case. Given that the input feature values are mostly continuous, the resulting clusters become numerous with most having single elements making this conjunctive categorization impractical. Furthermore, even for the few clusters that had more members, the responses still deviate from each other. This is why the use of a conceptual clusterer such as SOM becomes advantageous: it learns tractable polymorphous clusters while still maximizing both the members’ similarity and the clusters’ dissimilarity.

6. PERFORMANCE EVALUATION, ANALYSIS, AND FUTURE TASK

Predictive and actual RT values were correlated, and accuracy of RC prediction was computed as the percentage of hits (correct predictions) in total (hits+misses). Beck et al (2000) used correlation measure for the former since they were concern on whether response time will increase/decrease given the state of the tutorial instance. We compared the performance of MORPHUS vis-à-vis that of having no cluster knowledge, i.e., the perceptron and the naïve Bayesian classifier were used on the un-clustered dataset, in three experiment set-ups: (A) Complete – all instances in the dataset were used for training and testing, (B) Half-split (or two-fold) cross validation – predictive models were learned from half of the dataset, and the other half was used to test the model, and (C) 10-fold cross validation – learning from 90% of the data and testing on the remaining 10%. Table 2 shows the results of the experiments. The values in the last two rows are mean values (e.g., the average of doing 10 times the 90-10 split while varying the sets of training and test instances).

Table 2. Performance evaluation on MORPHUS

<table>
<thead>
<tr>
<th></th>
<th>No Clusters</th>
<th>MORPHUS</th>
<th>RC</th>
<th>No Clusters</th>
<th>MORPHUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Perceptron</td>
<td>Perceptron</td>
<td>Accuracy</td>
<td>Naïve Bayesian</td>
<td>Perceptron</td>
</tr>
<tr>
<td>Complete</td>
<td>0.560</td>
<td>0.655</td>
<td>Complete</td>
<td>69.2</td>
<td>72.5</td>
</tr>
<tr>
<td>Half-split</td>
<td>0.523</td>
<td>0.574</td>
<td>Half-split</td>
<td>68.1</td>
<td>71.7</td>
</tr>
<tr>
<td>10-fold</td>
<td>0.521</td>
<td>0.581</td>
<td>10-fold</td>
<td>69.1</td>
<td>72.1</td>
</tr>
</tbody>
</table>

The results show that the presence of cluster knowledge significantly improves RT prediction. MORPHUS is close to achieving a strong prediction for RT. Commonly, a 0.6-0.79 correlation is considered strong, especially for a very noisy variable such as time. On the other hand, though MORPHUS is accurate at the average when
predicting \( R_c \), there seems to be very little improvement given the cluster models. MORPHUS abstracts its techniques in a framework whereby cluster knowledge can be used to improve prediction; hence, its framework remains as change can be made to its current techniques. We suspect that the current clustering algorithm may have missed other possibly existing essential relationships in the data thereby failing to find the more “powerful” clusters and/or a non-linear approximation utilizing cluster knowledge may perform better. Though the perceptron is limited in its capability, choosing it for the meantime will pave the way for us to migrate to a powerful approximator whose basic concept is similar to the perceptron (e.g., multi-layered perceptron, radial basis function network). Either these or that at least for this dataset, cluster knowledge contribute very little in improving \( R_c \) prediction (not true for \( R_e \)) and/or student differences could be hardly captured in this binary-valued parameter, even if other techniques are used. For example, C4.5 (Quinlan, 1993), which is hailed in ML as a very powerful algorithm for predicting discretized values, performs similarly in a no-cluster \( R_c \) prediction (e.g., 74% for \( A \)-set-up, and 70.9% for \( B \)-set-up).

We also learned that the performance of MORPHUS depends on the size of the map and on the order of training instances. These are non-trivial issues. We ran another experiment while controlling these parameters. We started with 25% of the instances used for learning and 75% for testing (25-75 split). We added the next set of training data while keeping the previous set intact (50-50 and 90-10 splits). This was done while changing the map dimensions. We input the maximum desired length and width of the map and let SOM determine the final clusters. To keep all our experiments simple, we only tested for square maps (\( d \times d \) dimension, \( 2 \leq d \leq 10 \)). We have evidence that performance is best for maps that are \( 5 \times 5 \) or \( 6 \times 6 \). Table 3 shows the result from a \( 5 \times 5 \) map. In this last set-up, performance continues to improve as more data are mined and processed by MORPHUS. MORPHUS may start off with a workable amount of data (say, 50% of the possible scenarios) to achieve an initial prediction that is good at the average and continues to improve by updating its cluster knowledge and weights as succeeding recorded interaction data are incorporated. The results in Table 2 are also from a \( 5 \times 5 \) map except that we performed several runs for each set-up while varying the training and test data.

<table>
<thead>
<tr>
<th>Map Size</th>
<th>25-75 Split</th>
<th>50-50 Split</th>
<th>90-10 Split</th>
<th>100-0 Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_e ) Correlation</td>
<td>0.064</td>
<td>0.600</td>
<td>0.605</td>
<td>0.650</td>
</tr>
<tr>
<td>( R_c ) Accuracy</td>
<td>69.7, 67.1</td>
<td>72.0, 71.5</td>
<td>71.6, 72.4</td>
<td>72.4, 74.1</td>
</tr>
</tbody>
</table>

To address the influence of map dimension and order of instances, as well as finding the best techniques, are compelling and must be further solicited from the data mining community. It is conclusive, however, that there is potential in using polymorphous cluster knowledge to improve prediction of coarse-grained student information.

7. CONCLUSION

We leveraged on the capability of a conceptual clustering algorithm to discover structures in logged data of fine-grained student-ITS interactions, and used the formed clusters as predictive models for predicting high-level student information to support fine-grained level adaptation. The paper tries to answer the following questions:

1. How are we asking as we mine the data? Our aim is to determine if cluster knowledge can help improve predictions. Our cluster-based learning agent faced the difficulty imposed by the dataset, i.e., deal with a noisy parameter (response time) from a diverse source (learners). Nonetheless, our agent’s predictions are conditionally strong for this aspect. Predicting response correctness was more challenging in the fact that even the better techniques we tried fall short of more accurate predictions. Cluster knowledge seems to help very little to improve prediction here, albeit our agent’s performance remains accurate at the average.

2. How are the clusters characterized? We propose polymorphous clustering to model student and tutoring system’s interaction behaviours. In this way, interaction instances logged from the same student can be distributed to various clusters and learners differing in traits can co-exist in the same cluster. This means that our agent may give varying predictions for different cases even if the learner is the same. This form of case-based predictive modeling differs from the usual categorization in student modeling where only similar types of learners can co-exist in a group and each category is stereotyped with specified treatments.

It is the case that a conceptual clustering algorithm can discover structures in the data where clusters exhibit predictive characteristics. The next and more important issue is whether the cluster knowledge will be at all significant in improving prediction. Its prospect based on the results is conclusive.
REFERENCES


