Unsupervised and Supervised Machine Learning in User Modeling for Intelligent Learning Environments

Saleema Amershi[1] and Cristina Conati[1,2]

[1]Dept. of Computer Science,
University of British Columbia
2366 Main Mall, Vancouver, BC,
V6T 1Z4, Canada
{samershi,conati}@cs.ubc.ca

[2]Dept. of Information and Communication Technology
University of Trento,
Povo, Trento, Italy

ABSTRACT

In this research, we outline a user modeling framework that uses both unsupervised and supervised machine learning in order to reduce development costs of building user models, and facilitate transferability. We apply the framework to model student learning during interaction with the Adaptive Coach for Exploration (ACE) learning environment (using both interface and eye-tracking data). In addition to demonstrating framework effectiveness, we also compare results from previous research on applying the framework to a different learning environment and data type. Our results also confirm previous research on the value of using eye-tracking data to assess student learning.

ACM Classification: I5.5 [Pattern Recognition]: Implementation.-Interactive systems.
I6.5 [Simulation and Modeling]: Model Development.-Modeling methodologies
K3.1 [Computers and Education]: Computer Uses in Education.-Computer-assisted instruction

General terms: Human Factors, Experimentation

Keywords: User modeling, unsupervised and supervised machine learning, intelligent learning environments, eye-tracking

INTRODUCTION

In this research, we propose a user modeling framework that uses both unsupervised and supervised machine learning to address two of the most cited difficulties of developing user models for computer-based learning environments (e.g., [3, 11, 19]): laborious effort required by application designers to construct models, and limited model transferability across applications. The user model is a fundamental component of an *intelligent learning environment* (ILE), i.e., a computer-based system that can provides adaptive support for students. The user model guides the adaptation process by providing the ILE with an abstract representation of the learner in terms of relevant traits such as knowledge, meta-cognitive ability, and learning behaviors [3, 19].

Unfortunately, although the benefits of individualized computer-based instruction are well-recognized, so are the development costs, of which a considerable part is devoted to the user model [3]. This is especially true for *knowledge-based* user models, because they require eliciting the relevant domain and pedagogical knowledge from experts, a process that is often hard and time consuming. Furthermore, pure knowledge-based approaches can typically recognize and interpret only expected student behaviors, and are unable to handle unanticipated ones. Thus, they tend to be suboptimal for novel applications for which real experts do not exist yet.

To circumvent the drawbacks of knowledge-based student models, some researchers have turned to the field of machine learning (e.g., [2, 9]) to approximate functions that map observable student behaviors to classes such as the correctness of student answers. These functions can then predict the outcome of future student behaviors and inform adaptive facilities. However, this approach typically necessitates labeled data. When labels (e.g., student answers) are not readily available from the system, domain experts must resort back to manual labeling to supply them, which is again time-consuming and error prone.

The user modeling framework we propose addresses the issue of cost-intensiveness by integrating supervised and unsupervised machine learning. The framework is a generalization of the statistical pattern recognition approach [13] we used in [1] to automatically create a user model for an intelligent learning environment to teach AI algorithms. The general procedure for statistical pattern recognition is: data acquisition, processing,

IUI'07, January 28–31, 2007, Honolulu, Hawaii, USA.

learning, and then testing [13]. Our framework defines this process specifically in the context of user modeling. It uses unsupervised learning to automatically identify common learning behaviors and then applies supervised machine learning to these behaviors to train a classifier user model that can inform an adaptive ILE component.

A key distinction between our modeling approach and knowledge-based or supervised approaches with hand-labeled data is that human intervention is delayed until after unsupervised machine learning automatically identifies behavioral patterns. That is, instead of having to observe individual student interactions in search of meaningful patterns to model (e.g., student errors or misconceptions as in [5]) or to input to a supervised classifier (e.g., actions indicating motivation [9], instances of student's misusing an existing ILE [2]) the developer is presented with an unbiased picture of common behavioral patterns that can then be analyzed in terms of learning effects. Expert effort is further reduced by using supervised learning to actually build the user model from the identified patterns.

In addition to reducing developer workload, our approach also facilitates transfer across different applications and data types. In [1], we showed the effectiveness of the approach applied to logged student interface actions in an environment for teaching an AI algorithm, the CIspace CSP (Constraint Satisfaction Problem) Applet. Here, we demonstrate transferability by applying it to (i) a different learning environment, the Adaptive Coach for Exploration (ACE) for mathematical functions; (ii) data beyond interface actions, namely eye-tracking data. This data is higher dimensional than what we experimented with in [1] and so we extend our framework to include automatic feature selection to reduce data dimensionality.

Both the CIspace CSP applet and ACE are exploratory learning environments (ELEs), i.e. are designed to support learning via free, student-led exploration of the target domain. We chose ELEs as testbeds for our framework because previous research has shown the value of providing adaptive support for student exploration [18], but these environments are especially hard for traditional user modeling approaches. Because ELEs are a relatively novel learning paradigm, little practical knowledge exists about optimal learning strategies within these systems, increasing the difficulties of applying knowledge-based modeling approaches. Furthermore, because the space of possible interaction behaviors within ELEs can be very large, observing distinct behaviors and interpreting them in terms of learning effects is especially difficult. This makes supervised machine learning techniques that require manually labeled data also unappealing.

For instance, ACE's original student model [4], built via a fairly laborious knowledge-based approach, specified *which* components of the target domain (mathematical functions) should be explored for effective learning. However, due to lack of knowledge, it did not model *how* these components should be explored, leading to suboptimal model performance. A later version of the model [7] partially addressed this problem by using supervised learning to recognize effective exploration patterns from both interface actions and eye tracking data. But the relevant patterns had to be hand-labeled using an extremely laborious protocol analysis [7] and for this reason were limited to a subset of the available data. In this paper, we show that when applied to ACE, our framework can automatically identify more complex behavioral patterns than those identified by experts in [7]. An additional contribution is that fwe extend the results presented in [7] on the value of using eye-tracking data in modeling user reasoning processes.

In the rest of the paper, we first discuss related work. Next, we outline our user modeling framework. Then, we review previous results on applying our framework to the CSP Applet. Finally, we describe ACE and present the results of applying our framework to it. We conclude with a summary and suggestions for future work.

RELATED WORK

Unsupervised machine learning for user modeling has been mostly used in non-educational applications. For example, collaborative filtering (CF) systems employ unsupervised learning techniques to model user preferences and make item recommendations based on user similarities (e.g., [16]). Other research has demonstrated the use of unsupervised learning on (i) words in a document to model and automatically manage email activities [14]; (ii) frequencies of web pages access to automatically adapt web sites [17]. In contrast, research in using unsupervised machine learning for user modeling in educational systems remains rare [19]. A notable exception is MEDD [20] which uses unsupervised learning to discover novel classes of student errors and automatically build error libraries (for Prolog programming). Our approach differs from this in that we are modeling student interaction behaviors in unstructured learning environments instead of static student solutions and errors. [12] and [21] are also related to our work because, although they do not actually build user models, they also use pattern recognition approaches to discover patterns of student behaviors. DIAGNOSER [12], like MEDD, uses unsupervised learning to discover errors in static student solutions to physics questions. More similar to what we do, [21] uses clustering on interface action frequencies, to detect behavioral patterns in an environment for collaborative learning. Our work differs in that we use higher dimensional data including action latency, measures of variance and gaze information. In both [12] and [21] the resulting patterns are given to instructors who can then use them to tailor instruction, whereas we take this process one step further to automatically build a user model. Our research is also broader because we show transfer of our approach across applications and data types.

MODELING FRAMEWORK

Figure 1 shows the architecture of our proposed modeling framework, which divides the user modeling process in two main 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 used as input to an unsupervised learning (clustering) algorithm that groups them according to their similarity. These groups (clusters) represent students who interact similarly with the environment and are analyzed by the model developer to identify interaction behaviors as effective or detrimental for learning. In the online phase, the clusters identified in the offline phase are used directly in a classifier user model for learner classification. The online classifications and the learning behaviors identified by cluster analysis can then be used to inform an adaptive component that can encourage effective learning behaviors and prevent detrimental ones. In the rest of this section we describe the steps in each of the two phases, along with the algorithms we chose to complete these steps in the work presented here.

Offline Identification

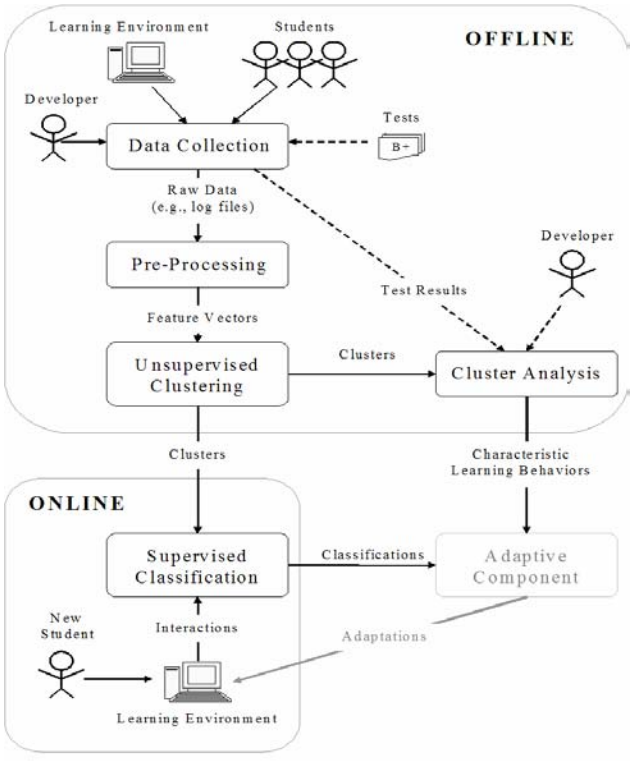

Figure 1: User modeling framework. Dotted lines represent optional input. Grayed out elements are outside of the framework.

Data Collection. The first step in the offline phase is to log data from students interacting with the target learning environment. Here, the developer requires knowledge (or a catalog) of all possible primitive interaction events that can occur in the environment so that they can be logged (see in Figure 1 the solid arrow from 'Developer' to 'Data Collection'). In addition to interface actions, logged data can include events from any other data source that may help reveal meaningful behavioral patterns (e.g., an eye-tracker).

An optional, but highly desirable additional form of data are tests on student domain knowledge before and after using the learning environment, to measure student learning with the system (see the dotted arrow in Figure 1 from 'Tests' to 'Data Collection'). These can then be used in cluster analysis, as we will see below.

Preprocessing. Clustering operates on data points in a feature space, where features can be any measurable property of the data. So in order to find clusters of students who interact with a learning environment in similar ways, each student must be represented by a (multidimensional) feature vector. The second step in the offline phase is to generate these vectors by computing low level features from the data collected. We suggest features including (a) the frequency of each interface action, and (b) the mean and standard deviation of the latency between actions. The latency dimensions are intended to measure the average time a student spends reflecting on action results, as well as the general tendency for reflection (e.g., consistently rushing through actions vs. showing selective attention). In the current research we also include features extracted from eye-tracking data (i.e., eye gaze movements).

In high-dimensional feature spaces, natural groupings of the data are often obscured by irrelevant features. Therefore, determining the most salient features and removing irrelevant ones (called *feature selection*) can significantly improve results of the subsequent machine learning algorithm. We suggest using an entropy-based unsupervised algorithm [8] for feature selection. First, we rank each of the candidate features according to the entropy (disorder) induced by the removal of that feature from the entire set of data points. Next, we run forward selection on the ranked features. This evaluates incrementally larger feature subsets in terms of their performance in clustering and chooses the subset that maximizes cluster quality, defined as having maximum between-cluster variance and minimum within-cluster variance. Note that feature selection must execute clustering on each candidate subset in order to assess cluster quality, and returns both a reduced feature set and the resultant clusters. Thus, clustering need not be performed again when using this automatic feature selection technique. When no feature selection is performed then clustering must still be carried out to determine behavioral patterns, as described next.

Clustering. Clustering works by grouping feature vectors by their similarity, where here we define similarity as the Euclidean distance between data points in the normalized feature space [10]. For both the work presented in this paper and that reported in [1], we chose a popular partition-based algorithm for clustering called k-means [10]. K-means takes as input feature vectors and a user-specified k value specifying the number of clusters that should be returned. Initially, k data points are randomly selected to be the cluster centroids. The remaining data points are then assigned to the cluster whose current centroid minimizes the Euclidean point-to-centroid distance. After all data points are assigned to a cluster, new cluster centroids are computed from these groupings. The process then repeats for a given number of iterations or until there are little or no changes in the clusters. K-means can often converge at local optima depending on the selection of the initial cluster centroids. Thus, several trials are typically executed to find high quality clusters.

While *k*-means is efficient for large data sets, and so may be favorable for online educational technologies that have the potential to log large amounts of data, it does have limitations. First, *k*-means assumes the clusters are elliptical, and would be unsuccessful at identifying more complex cluster shapes. In this case, a more computationally expensive hierarchical algorithm may be best [10]. *K*-means also produces hard assignments of data points to clusters, whereas it may be beneficial for an adaptive ELE to know the uncertainty in the assignments. Here, a probabilistic version of *k*-means called Expectation Maximization [10] may be more appropriate. So the choice of clustering algorithm should be informed by properties of the data or the application being studied. Although we use *k*-means as proof of concept throughout this research, we expect that other clustering algorithms can be substituted for *k*-means in our framework.

Cluster Analysis. If the clusters detected by clustering are to be used in a user model for an ILE, the clusters must be analyzed and interpreted to determine which patterns of behaviors are effective or ineffective for learning. This is best done by using objective information about learning gains from application use, e.g., improvements from pre to post tests, to identify which clusters of students were successful learners and which were not (see dotted arrow marked 'Test Results' between 'Data Collection' and 'Cluster Analysis' in Figure 1). If learning gains are unknown, then expert evaluation is required to interpret the cluster characteristics in terms of learning (illustrated in Figure 1 by the dotted arrow from 'Developer' to 'Cluster Analysis'). In this case, human workload is still reduced because they avoid the time-consuming process of having to observe individual student interactions and then look for meaningful patterns.

An additional step in cluster analysis is to evaluate clusters for similarities and dissimilarities along each of the feature dimensions in order to characterize the different learning behaviors. While this step is not strictly necessary for on-line recognition based on supervised learner classification, it can be useful to help developers gain insights on the relevant learning behaviors and devise accurate adaptive interventions targeting them.

In this research, we use formal tests to compare clusters in terms of learning and distinctive interaction behaviors. To compare the clusters obtained with k=2, we use Welch's t-tests (Student's t-test corrected for unequal sample variances) to determine the statistical significance of the differences (throughout the paper, we use .05 for significance and .1 for marginal significance). We also measure effect sizes (the magnitude of the differences), using Cohen's d [6], to determine the practical significance of the differences. We consider a large effect ($d > .8$) to be significant and a medium effect ($.8 > d > .5$) to be marginally significant as per Cohen's standard. When k>2, we use one-way analysis of variances (ANOVAs) with Tukey HSD adjustments for post-hoc pairwise comparisons to determine statistical significance.

Online Recognition

Supervised Classification. The second phase supported by our modeling framework (lower left of Figure 1) uses an online supervised classification algorithm to recognize effective and ineffective learners by classifying a new student into the distinct learner groups found by offline clustering. This is done via an online *k*-means classifier that incrementally updates the classification of a new student into one of these groups as the student interacts with the learning environment. As actions occur, the feature vector representing the student's behavior thus far is updated to reflect the new observation. Next, the student's classification is computed by simply recalculating the distances between the updated vector and each cluster centroid and then assigning the feature vector to the cluster with the nearest centroid.

PREVIOUS RESULTS ON USING THE FRAMEWORK WITH THE CSP APPLET

In [1], we applied our framework to model student interactions with the CIspace CSP Applet, an ELE to support learning of an AI algorithm for constraint satisfaction problems (CSP). The CSP applet uses visualizations to dynamically demonstrate the workings of the algorithm. Students are free to explore these visualizations using any of seven different functions embedded in the interface. Thus, our data set consisted of 21-dimentional feature vectors representing action frequencies of these seven functions as well as the means and standard deviations of the latency between actions. In that work we showed how the unsupervised component of our modeling framework was able to identify student clusters (for k=2 and k=3) characterized by significantly different interaction behaviors (as well as significantly different learning outcomes), and several of these behaviors would have been difficult to recognize and label by hand. The model developed, with k=2, through

the supervised learning component achieved a good overall predictive accuracy of 88.3% on new student behaviors, although the accuracy was higher for ineffective behaviors than for effective behaviors because of the limited data available for the latter. In contrast, the overall predictive accuracy of the model developed with k=3 was only 66.2%. This is likely attributable to the smaller cluster sizes resulting from the larger k value in this case. Like with the model built with k=2, the accuracy for the model developed with k=3 was highest for the largest cluster which was again characterized by ineffective learning behaviors. We now discuss the results of applying the framework to a different system, the ACE ELE for mathematical functions, and to a broader set of data types including both interface actions and eye-tracking information.

ADAPTIVE COACH FOR EXPLORATION (ACE)

ACE [4] is an intelligent ELE that provides tools to support student-led exploration of mathematical functions while an adaptive Coach provides tailored suggestions on how to improve student exploration. ACE is comprised of three units, each designed to present concepts pertaining to mathematical functions in a distinct manner. In this research we focus on the *Plot Unit* (Figure 2) because it offers the widest range of exploratory activities, making it an ideal candidate to evaluate our modeling framework.

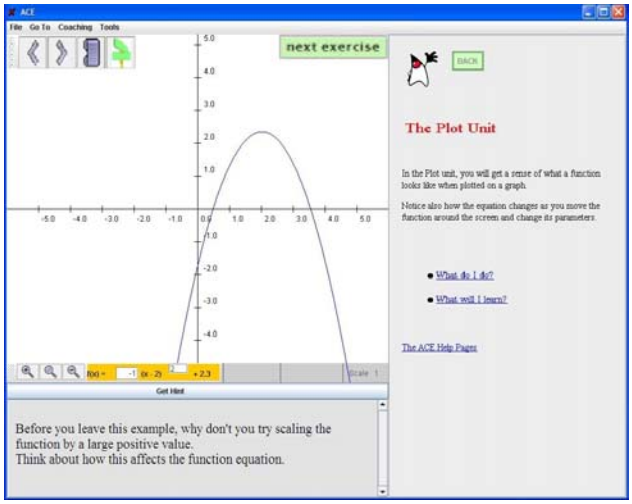

Figure 2: The ACE Plot Unit.

ACE's Plot Unit provides three types of functions, or exercises for the student to explore: constant, linear and power functions. Each function has an associated set of exploration cases that together illustrate the full range of function attributes. For example, linear functions are defined by two parameters specifying the function slope and the y-intercept. In order to gain a broad understanding of linear functions, the student should study the relevant exploration cases, including positive and negative intercepts, and positive, negative and zero slopes.

The Plot Unit interface (Figure 2) is divided into three components. Function exploration occurs within the top-left panel, which visually demonstrates the relationship between mathematical equations and their corresponding plots in a Cartesian plane. The component at the right contains hypertext help pages about ACE's interface and the functions that can be explored. The lower left panel is where ACE's Coach displays tailored, on-demand hints to guide the student's exploration. In addition, the Coach intervenes (via a dialog box) if the student tries to move to a new exercise before sufficiently exploring the current one. In this situation the Coach tries to encourage the student to continue exploring the current exercise by offering them a hint if they stay. However, in keeping with the theme of student-controlled exploration, the student ultimately decides whether or not to move on.

To explore mathematical functions, ACE allows students to take the following 13 interface actions:

- *Plot Move* (*PM*) – Dragging the function plot around the screen. The parameters of the function's equation (directly below the Cartesian plane in Figure 2) are automatically adjusted to reflect the transformation.
- *Equation Change* (*EC*) – Editing the function equation. ACE transforms the function plot accordingly.
- *Reset* – Resetting the function to its initial parameters.
- *Next Exercise* (*NE*) – Stepping sequentially forward to the next exercise in the pre-defined curriculum by clicking on the *NE* button (top right of plot).
- *Step Forward* (*SF*) – Same as *NE* action, but done by clicking on the forward arrow on the toolbar at the top right of the plot.
- *Step Back* (*SB*) – Stepping backwards to the previous exercise.
- *Lesson Browser* (*LB*) – Opening the *LB* tool which outlines the curriculum and allows the student to jump to any exercise within the curriculum.
- *Exploration Assistant* (*EA*) – Opening the *EA* tool which displays the exploration cases already examined by the student and remaining to be examined.
- *Get Hint* – Requesting a hint from the Coach.
- *Stay* - Adhering to the Coach's advice to continue exploring the current exercise.
- *Move On* (*MO*) – Ignoring the Coach's advice to stay on the current exercise and moving on to another one.
- *Help* – Using the hypertext help pages.
- *Zoom* – Zooming into or out of the graph region.

The Coach's interventions are guided by a knowledge-based user model [4]. The model is a hand-constructed Dynamic Bayesian Network that includes nodes to represent all possible exploration cases, nodes to represent student's understanding of related mathematical concepts

and links representing how exploration of relevant cases relate to concept understanding. To assess whether a case has been explored effectively, the network includes information on both student actions (only the *PM* and *EC* actions described above) and time elapsed between these actions. The latter is used as an estimate of student's active reasoning on each exploration case. The network parameters (i.e., multi-valued conditional probability tables for each node) were manually defined using prior knowledge or estimations.

USING THE FRAMEWORK WITH ACE

Data Collection

The data we apply our modeling framework to was obtained from a previous user study involving the ACE Plot Unit [7]. The goal of the study was to analyze if and how eye-tracking data on gaze patterns helps assess student reasoning on individual exploration cases and consequent effectiveness of student exploration (student reasoning was assessed solely from latency between actions in the original ACE model). 36 students participated in the user study. They first took a pre-test on mathematical functions, and then interacted with ACE for as long as they needed. While using ACE, the students were asked to verbalize all of their thoughts. Student gaze was tracked by a head-mounted eye-tracker. In addition, all student interactions with ACE were logged and synchronized with data from the eye tracker. Finally, the students took a post-test similar to the pre-test. For the research presented here, we obtained 3783 interface actions over 673.7 minutes from the study log files, along with the accompanying gaze data from the eye-tracker.

Data from this study was also used in [7] to build a new version of ACE student model using supervised machine learning. This new model uses gaze information, in addition to latency between actions, to assess effectiveness of student exploration. The use of gaze information is limited to gaze shifts between the plot and equation area after a plot move or equation change, which intuitively should indicate student's reflection after these actions. Although gaze pattern information may also be relevant in relation to other interface actions, this work was limited to equation and plot changes because of the effort required to generate the hand labeled data necessary to train the model. Two researchers (to assure coding reliability) categorized student verbalizations after equation and plot changes as instances of student reflection vs. speech not conducive to learning. Then, they mapped them onto presence/absence of gaze shifts and latency until the next action. This new model showed better performance in assessing effectiveness of student exploration than models using only action occurrences or action occurrences plus latency information, showing the value of eye-tracking data for this type of assessment. In the following two sections, we compare the results of applying our framework to the data described above, with the results obtained by the supervised approach in [7].

Preprocessing and Unsupervised Clustering

We extracted two different sets of features from the ACE study data. The first set (FeatureSet1) consisted only of interface features, i.e. frequencies of each of the 13 possible interface actions and the mean and standard deviation of the latency between actions. This feature set is analogous to the one used in [1], so as to evaluate how our modeling framework transfers across different applications using the same type of input data.

The second set (FeatureSet2) included features distilled from the eye-tracking data in addition to the above interface features. We chose this set for two reasons. First, we wanted to evaluate how our approach works on a range of different data sources. Second, we wanted to see if we could reproduce results in [7], showing that eye-tracking information improves assessment of the effectiveness of student exploration. In particular, we hypothesized that eye-tracking data would improve the performance of clustering in identifying groups of students with distinct learning proficiency. We focused on the two gaze shift patterns used in [7]: direct and indirect gaze shifts. A direct gaze shift happens when a student's gaze moves directly between the function equation and its plot, while in an indirect gaze shift, gaze moves to non-salient regions in between. Although in [7] the authors only considered these gaze shifts after plot moves and equation changes, they may be relevant after most ACE interface actions. For example, after a *next exercise* action a new function appears on the screen requiring attention to both the plot and equation regions in order to understand the connection between the new function equation and its plot. Since, contrary to the supervised approach in [7], considering more actions in our approach does not involve much extra work, we included gaze shift information for all the 13 interface actions in FeatureSet2, by computing the mean and the standard deviation of the number of indirect and direct gaze shifts as additional features.

FeatureSet1 and FeatureSet2 included respectively 39 and 91 possibly influential features. With only 36 feature vectors corresponding to the 36 study participants, these high-dimensional feature spaces can result in data sparseness and may degrade the performance of clustering. Therefore, as outlined in our modeling framework, we performed entropy-based feature selection on each set. We used k set to 2, 3 and 4 for the k-means clustering executed during forward selection. We chose these values because our data set was relatively small and so we only expected to find a few clear groups with distinct learning outcomes.

Because of space limitation, here we only discuss results from feature selection on FeatureSet2 with k=2, the only case in which we found significant differences in student learning outcomes of the obtained clusters. 36 of the 91 original features in FeatureSet2 were selected as important (listed in Table 1). All action frequencies are

selected as important, except in the case of a *Stay* action. Gaze shift dimensions are only identified as important in the presence of the corresponding latency dimensions (see *EC* and *NE* entries in Table 1 for example). Conversely, latency was found to be relevant independently of gaze shift features, for instance in relation to using the ACE *help* pages (see *help* entries in Table 1). This agrees with the findings in [7] that gaze shifts may be important mostly in discriminating between time spent reflecting on an action's results and idle time.

Feature Description	HL average	LL average	*p*	*D*
PM freq.	.024	.034	.116	.418
EC freq.	.015	.019	.203	.305
***EC* latency avg**.	21.8	16.1	.047*	.677
EC latency sd.	10.4	6.08	.073	.636
***EC* indirect avg.**	1.21	.440	.012*	1.02*
***EC* indirect sd.**	1.12	.556	.022*	.886*
***Reset* freq.**	0	.001	.008*	.735
Reset latency sd.	0	.051	.082	.406
***NE* freq.**	.005	.009	.005*	.827*
***NE* latency avg.**	18.7	13.2	.003*	1.07*
NE latency sd.	10.3	7.44	.059	.594
***NE* indirect avg.**	1.74	.625	1e-5*	2.18*
***NE* indirect sd.**	2.09	.715	1e-4*	1.97*
***NE* direct avg.**	1.30	.201	.003*	1.41*
***NE* direct sd.**	1.79	.362	.006*	1.25*
***SF* freq.**	.008	.011	.034*	.621
***SF* latency avg.**	7.65	4.65	.008*	1.03*
***SF* latency sd.**	9.96	5.83	.014*	.858*
SB freq.	0	2e-4	.122	.338
SB latency avg.	0	.400	.053	.475
SB indirect avg.	0	.040	.164	.283
***LB* freq.**	0	6e-4	.037*	.528
***EA* freq.**	2e-4	.001	.018*	.640
EA latency avg.	5.27	5.65	.472	.027
Get hint freq.	3e-4	5e-4	.312	.155
***Stay* latency avg.**	6.55	2.54	.032*	.775
Stay indirect avg.	.152	.100	.350	.136
***MO* freq.**	.003	.005	.014*	.744
MO latency avg.	2.23	232	.163	.283
MO latency sd.	2.76	400	.163	.283
Help freq.	.002	.001	.234	.288
***Help* latency avg.**	2.45	7.28	.030*	.587
***Zoom* freq.**	4e-4	.021	.008*	.741
***Zoom* latency avg.**	.374	1.98	.003*	.961*
***Zoom* latency sd.**	.700	2.70	.017*	.785
Zoom direct sd.	0	.101	.012*	.683

* Significant at $p<.05$ or $d>.8$ (feature description in bold)

Table 1. Pairwise comparisons between HL and LL clusters along the features selected from FeatuteSet2

Interestingly, neither latency nor gaze shifts were found to be relevant after a *plot move* (see *PM* table entry). Given that both *plot moves and equation changes* are exploratory actions requiring student reflection, this result appears unintuitive, especially considering that latency and gaze shifts were found to be important after *equation change*s. This could be an artifact of forward selection, which may prematurely rule out certain features that are important only in combination with features not yet included in the subset (i.e. lower ranked features). However, since clustering was in fact able to find distinct learner groups using only the features returned by feature selection, these findings could challenge our previous beliefs about the utility of *plot move* actions for learning.

Cluster Analysis

As dictated by our framework, in this phase we first compare the clusters returned by feature selection on FeatureSet1 and FeatureSet2 in terms of student learning gains (derived from the pre and post-test scores available from the user study described earlier). When significantly different learning gains were found, we then compared the clusters in terms of differences in behavioral patterns.

Cluster Analysis for FeatureSet1. With FeatureSet1, for all values of k we found no significant differences in learning gains amongst clusters and so we cannot use the clusters as the basis for the on-line modeling phase. Interestingly, in [1] we were able to find distinct learner groups by using only interface actions on a data set comparable in size to the data set we are using here. We hypothesize that this discrepancy is due to differences in the nature of the domains and interfaces of the two learning environments. The AI algorithm that the CSP Applet is designed to demonstrate is more complex compared to the relationship between mathematical functions and their graphs that the ACE Plot Unit demonstrates. As a result, the CSP Applet interface includes several functions that allow the student to visualize and reflect on the workings of the AI algorithm, whereas ACE only provides two such functions: plot moves and equation changes. Thus with the CSP Applet, interface actions alone may capture student reflection during exploration better than interface actions alone in ACE. This hypothesis is consistent with the results in [7] showing that gaze patterns, together with action latency, predict student reflection and learning better than sheer number of actions or action latency alone. Additional data may be necessary [13] to detect distinct learner groups using only this feature set.

Cluster Analysis for FeatureSet2. With FeatureSet2 and k=2 we found a marginally significant difference in learning gains between the two clusters returned ($t(17.85)=1.55$, $p=.069$, $d=.571$). Furthermore, several of the clusters' distinctive behaviors involved gaze patterns, as we discuss next. This shows that incorporating eye-tracking data into feature vectors improves the performance of clustering in identifying groups of

students with distinct learning proficiency, as compared to using interface actions and latency information alone.

Hereafter we refer to the group with high and low average learning gains as the 'HL' and 'LL' groups respectively. In order to characterize these two learner groups in terms of interaction behaviors, we did a pair-wise analysis between the clusters on each of the 36 feature dimensions. Table 1 presents the results of this analysis. Here we discuss some of the most interesting findings.

Some of these findings are consistent with results in [7], as we were hoping. First, there were no statistically significant differences in the frequency of *plot moves* or *equation changes* between the HL and LL groups, consistent with finding in [7] that sheer number of exploratory actions is not a good predictor of learning in this environment. Second, after an *equation change*, the LL group would pause for a significantly shorter duration than the HL group on average (see '*EC* pause avg.' in Table 1). In [7], the authors determined 16 seconds to be an optimal threshold between occurrences of effective reflection on exploration cases and other verbalizations not conducive to learning. Consistent with this result, Figure 3 shows that the average latency by the students in the HL group were mostly above this threshold, whereas with the LL group the latency averages were centered about the threshold.

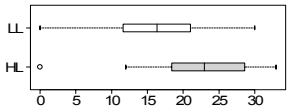

Figure 3: Boxplot of avg. latency after *Equation changes* between HL (gray) and LL (white) clusters.

Because with clustering we are able to incorporate all interface actions and associated gaze data simply by including them in the multi-dimensional feature vectors, we also found patterns additional to the ones found in [7]. For example, the students in the HL group were more varied in how often they would indirectly gaze shift after an *equation change* (see '*EC* indirect sd.' in Table 1). This selective behavior suggests that students need not reflect on the results of every exploratory action in order to learn well so long as they do not consistently refrain from reflection. In addition, the LL group paused less and made significantly fewer indirect gaze shifts after an *equation change* than the HL group (see '*EC* indirect avg.' in Table 1). These results are consistent with less reflection by the LL group compared to the HL group and may account for some of the difference in learning gains. It should be noted that in [7], individual gaze shifts, not multiple gaze shifts, were found to predict student reasoning. In that research, gaze behavior was studied only in the context of plot moves and equation changes because of the effort of labeling data. The fact that we are using all interface actions and accompanying gaze data may account for this discrepancy in using multiple gaze shifts. We found similar differences in the latency and gaze shifting behaviors of the two groups when a new function appeared on the screen after a *next exercise* action (see *NE* latency and gaze entries in Table 1).

When the Coach suggested that the student spend more time exploring the current exercise, LL students chose to ignore the suggestion and *move on* to another exercise significantly more frequently than HL students (see '*MO* freq.' in Table 1). This result is intuitive since the Coach's suggestions are intended to promote effective learning [4] and so ignoring them would be expected to adversely affect students. The frequency of *Stay* actions were not found to be relevant by feature selection, however when they did occur, HL students paused for significantly longer than LL students (see '*Stay* latency avg.' in Table 1). This is another intuitively good behavior, possible showing that the HL students followed the Coach's advice more carefully by spending additional time pondering over the current exercise before taking additional actions.

While the above patterns are quite intuitive, this approach was also able to identify additional patterns that do not have an obvious relation to learning. For example, the LL students advanced sequentially through the curriculum using the *next exercise* and *step forward* buttons significantly more frequently than the HL group (see '*NE* freq.' and '*SF* freq.' in Table 1). Considering that every student examined all three available exercises, intuition would suggest that there should be no differences between the clusters along these dimensions. However, further examination of the clusters reveals that the LL students also made use of both the *step back* feature and the *Lesson Browser* tool to navigate through the curriculum, whereas none of the HL students performed these actions. Since the LL students showed lower learning gains after interacting with ACE, it is probable that these students were moving impulsively back and forth through the curriculum. This hypothesis is substantiated by the fact that the Coach's suggestion to continue exploring the current exercise (computed by combining the frequencies of *move on* and *Stay* actions) appeared more frequently ($t(25.66)=1.57$, $p=.063$, $d=.536$) to the LL students than to the HL students. As this pattern involved several interface features (i.e., *NE*, *SF, SB, LB, MO* and *Stay*) it may have been difficult to observe, even by application experts.

Similarly, there were unintuitive differences in the use of the *zooming* features between the two groups (see '*zoom'* features in Table 1). The LL students zoomed into or out of the plot region significantly more frequently than the HL students. The HL group students paused for a consistently shorter duration after zooming than the LL students on average. Although zooming may not have clear pedagogical benefits, this behavior may suggest confusion on the part of the LL students resulting in the need for more detailed inspection of the plot. This is consistent with the finding related to *help* page

exploration. Here, LL students paused for significantly longer after navigating to a *help* page then the HL students (see '*Help* latency avg.' in Table 1) indicating that these students may have felt confused about how to use ACE or about the domain concepts and so required more help than the HL students.

Supervised Classification

In this section we first describe the method we used to evaluate our *k*-means-based online classifier. Next, we present the results on the performance of the model at recognizing students interacting with ACE as belonging to the LL or HL clusters identified in the offline phase.

Time restrictions prevented us from running additional user studies to collect test data for our model. Therefore, we performed a 36 fold leave-one-out cross validation (LOOCV) evaluation to make use of the available data and provide initial evidence of the online classifier's accuracy. In each fold, one student's data was removed from the training set, and the reduced set was re-clustered by *k*-means. Then, the removed student's data was fed into a classifier model trained on the reduced data set, and online predictions were made for the incoming actions as explained in the *Online Recognition* section of our framework description. Model accuracy is evaluated as student actions are observed (over time), where accuracy is measured as the percentage of students correctly classified into the clusters to which they were assigned in the offline phase.

It should be noted, however, that by using a LOOCV strategy, we run the risk of altering our initial cluster characterizations derived in the offline phase using the entire data set. Therefore, we should not expect to achieve 100% accuracy even after seeing all the actions performed by a student, because we are classifying incoming data using the clusters found by the reduced data set given by LOOCV. This issue is known as *hypothesis stability* [15]. Thus, prior to assessing predictive accuracy of our online classifier, we estimate the stability cost, or the difference between the clusters produced by LOOCV and the original clusters, as in [15]. Low stability cost helps to ensure that our model is essentially predicting what we would like it to predict, i.e., the membership of the removed student's behavioral patterns in one of the learning groups in the offline phase. The estimated stability cost for our *k*-means classifier model was 0.062 (where 0 is considered perfect stability and 1 is considered maximum instability [15]). This means that the characteristic behaviors of the two clusters identified in the offline phase are reasonably preserved during our LOOCV evaluation.

Figure 4 shows the percentage of correct predictions as a function of the percentage of student actions seen by the *k*-means online classifier model over time. The accuracy of the model converges to 97.2% after seeing all of the students' actions. Averaged over time, the accuracy is 86.3%. The figure also shows the model's performance over both the HL and LL groups. The accuracy for the LL group remains relatively stable over time, whereas the performance for the HL group is initially poor but increases to over 80% after seeing about 45% of the actions.

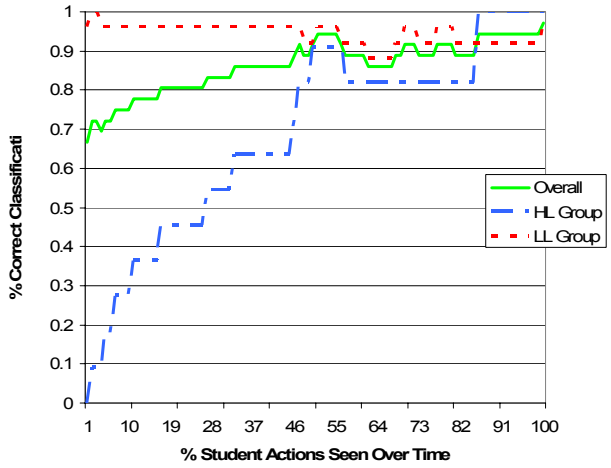

Figure 4: *K*-means classifier performance over time.

The LL and HL group accuracies effectively measure the sensitivity (i.e., the ability to detect suboptimal learning behaviors when student learning gains are poor) and specificity (i.e., the ability to detect effective learning behaviors when student learning gains are indeed high) of the classifier respectively. Table 2 shows these accuracies averaged over time (i.e., by taking the average of all of classification accuracies computed while students interacted with ACE over time).

	ACE	CSP (*k*=2)	CSP (*k*=3)
Accuracy	86.3%	88.3%	66.2%
Sensitivity (True Positive Rate)	94.2%	93.5%	66.1%
Specificity (True Negative Rate)	68.3%	62.4%	63.3%

Table 2. Classification accuracies averaged over time.

For comparison with previous framework application, Table 2 shows the similar results we obtained for *k*=2, and the results we obtained for *k*=3, by applying our framework to the CIspace CSP Applet data. The high sensitivity rates obtained for *k*=2 in both framework applications mean that these models would be useful for recognizing when a student behaves in ways ineffective for learning, essential for providing adaptive support for students who do not learn well with a learning environment. The sensitivity rate reported in the table for *k*=3 on the CIspace CSP Applet data was computed by combining the accuracy results for the two groups that showed ineffective learning behaviors in this case. The individual accuracies for these two groups were 80.9% and 44.9% averaged over time. Therefore, this model

would be most useful for recognizing students behaving in the ineffective ways characterized by the first (larger) group, but not by the second (smaller) group. Within an exploratory setting where student control is key, the low specificity rates for all of the models may cause an adaptive support system to interfere with an HL student's natural learning behavior if it is sometimes suboptimal. This imbalance is likely due to the distribution of the sample data [22] in all cases. For ACE, the HL group had fewer data points than the LL group (11 compared to 25), and similarly for the CSP Applet dataset for k=2. For k=3 on the CSP Applet dataset, the largest group (of ineffective learners) showed the highest accuracy, whereas the two smaller groups (one other group of ineffective learners and one group of effective learners) showed lower accuracies. This is a common phenomenon observed in classifier learning. Collecting more training data to correct for this imbalance, even if the cluster sizes are representative of the natural population distributions, may help increase the specificity rates of the models [22].

CONCLUSION AND FUTURE WORK

In this research, we presented a user modeling framework that makes use of both unsupervised and supervised machine learning in order to reduce the development costs typically associated with knowledge-based approaches to user modeling and supervised approaches that require hand-labeled data. Results of applying the framework to the ACE ELE confirm results in [7] on the value of eye-tracking data in revealing student reflection. And perhaps more interestingly, our approach was able to identify more complex patterns than was found through observation in [7]. In addition, we have demonstrated framework transferability across applications by comparing results with previous results on the CIspace CSP Applet ELE in [1].

Our next step is to collect more training data to see if this would help reveal clusters using only interface actions, as with the CIspace CSP Applet data, and testing data to better evaluate our user model. We also intend to build an adaptive component for ACE that uses the model built via our modeling process. The effectiveness of an adaptive ACE that uses our model could then be evaluated in a real world setting.

REFERENCES

1. Amershi, S. and Conati, C. Automatic Recognition of Learner Groups in ELEs. In *ITS* (Jhongli, Taiwan), 2006, pp. 463-472.
2. Baker, R.S., Corbett, A.T., and Koedinger, K.R. Detecting Student Misuse of ITSs. In *ITS* (Maceio-Alagoas, Brasil), 2004, pp. 531-540.
3. Beck, J., Stern, M., and Haugsjaa, E., Applications of AI in Education. *ACM Crossroads 3,* 1 (1996), 11-16.
4. Bunt, A. and Conati, C., Probabilistic Student Modeling to Improve Exploratory Behavior. *UMUAI 13,* 3 (2003), 269-309.
5. Burton, R.R. DEBUGGY: Diagnosis of Errors in Basic Mathematical Skills. In *ITS,* 1982, pp. 157-183.
6. Cohen, J., *Statistical Power Analysis for the Behavioral Sciences*. 2 ed. Lawrence Earlbaum Associates, Hillsdale, 1988.
7. Conati, C. and Merten, C., Gaze-Tracking for User Modeling in Intelligent Learning Environments: an Empirical Evaluation. To Appear in *Knowledge Based Systems, Special Issue on Techniques and Advances in Intelligent User Interfaces*.
8. Dash, M. and Liu, H. Feature Selection for Clustering. In *PACKDDM* (Kyoto, Japan), 2000, pp. 110-121.
9. de Vicente, A. and Pain, H. Informing the Detection of the Students' Motivational State: An Empirical Study. In *ITS* (Biarritz, San Sebastian), 2002, pp. 933-943.
10. Duda, R.O., Hart, P.E., and Stork, D.G., *Pattern Classification*. 2 ed. Wiley-Interscience, NY, 2001.
11. Gorniak, P.J. and Poole, D. Building a Stochastic Dynamic Model of Application Use. In *UAI* (Stanford, CA), 2000, pp. 230-237.
12. Hunt, E. and Madhyastha, T. Data Mining Patterns of Thought. In *AAAI Workshop on Educational Data Mining* (Menlo Park, California), 2005, pp. 31-39.
13. Jain, A.K., Duin, R.P.W., and Mao, J., Statistical Pattern Recognition: A Review. *IEEE Pattern Analysis and Machine Intelligence 22,* 1 (2000), 4-37.
14. Kushmerick, N. and Lau, T. Automated Email Activity Management: An Unsupervised Learning Approach. In *IUI* (San Diego, California), 2005, pp. 67-74.
15. Lange, T., Braun, M., Roth, V., Buhmann, J., Stability-Based Model Selection. In *NIPS* (Whistler, BC), 2003, pp. 617-624.
16. Paliouras, G., Papatheodorou, C., Karkaletsis, V., Spyropoulos, C., Discovering User Communities on the Internet Using Unsupervised ML Techniques. *Interacting with Computers 14,* 6 (2002), 761-791.
17. Perkowitz, M. and Etzioni, O., Towards Adaptive Web Sites: Conceptual Framework and Case Study. *AI 118,* 1-2 (2000), 245-275.
18. Shute, V.J., *A Comparison of Learning Environments: All That Glitters...* in *Computers as Cognitive Tools*, S.P. Lajoie and S.J. Derry, Editors., Lawrence Erlbaum Associates: Hillsdale, NJ, 1993.
19. Sison, R. and Shimura, M., Student Modeling and Machine Learning. *AIED 9,* (1998), 128-158.
20. Sison, R., Numao, M., and Shimura, M., Multistrategy Discovery and Detection of Novice Programmer Errors. *ML 38,* (2000), 157-180.
21. Talavera, L. and Gaudioso, E. Mining Student Data to Characterize Similar Behavior Groups in Unstructured Collaboration Spaces. In *Workshop on AI in CSCL* (Valencia, Spain), 2004, pp. 17-23.
22. Weiss, G.M. and Provost, F., The Effect of Class Distribution on Classifier Learning: An Empirical Study. Rutgers Univ., 2001.