A Data-Driven Approach for Inferring Student Proficiency from Game Activity Logs

Mohammad H. Falakmasir[⋆,#] **José P. González-Brenes**[⋆] **Geoffrey J. Gordon**[§] **Kristen E. DiCerbo**[⋆]

[⋆]School Research
Pearson
{jose.gonzalez-brenes,
kristen.dicerbo}@pearson.com

[#]Intelligent Systems Program
University of Pittsburgh
falakmasir@pitt.edu

[§]Machine Learning
Carnegie Mellon University
ggordon@cs.cmu.edu

ABSTRACT

Traditional assessments are important in education because they allow collecting evidence about student progress. Unfortunately, they can be very tedious to their stakeholders. In contrast, invisible assessment unobtrusively gathers performance data from students' daily activities to make inference about student relevant competency. We present a novel data analysis pipeline, Student Profficiency Inferrer from Game data (SPRING), that allows modeling game playing behavior in educational games. Unlike prior work, SPRING is a fully data-driven method that does not require costly domain knowledge engineering. We validate our method using data collected from students playing 11 educational mini-games. Our results suggest that SPRING is accurate to predict Math assessments (R^2 =0.55 , Spearman ρ=0.82).

ACM Classification Keywords

K.3.1 Computer Uses in Education.

Author Keywords

Educational Games, Student Modeling, Stealth Assessment

1. INTRODUCTION

Educational assessments are important because they collect evidence about about whether the teaching goals are being met. Unfortunately, the process of administering assessments is usually disconnected to the learning environment, and it is often disruptive to the classroom. In many developed countries, students now find themselves spending increasing amounts of time preparing and taking tests instead of learning [7]. For example, a survey of the current state of testing in America revealed that students are taking an average of 113 standardized tests between pre-K and highschool [10]. For these reasons, it is not surprising that the recent political climate and the general population have been weighing in on the question of whether students are being tested too much [10].

According to Evidence Centered Design (ECD) [15], the goal of assessment is to characterize the strength of evidence regarding claims one wants to make about individuals or groups. Therefore, the assessment process involves identifying, organizing, or creating activities for students so we may observe that evidence. An interesting alternative to traditional summative assessment is invisible (or stealth) assessment [18], where the evidence is gathered from leaners unobtrusively from the digital interactions of their ongoing activities. This data is used to understand claims regarding what students know and what they can do [19]. Stealth assessment is also intended to reduce or eliminate test anxiety, while not sacrificing validity and reliability [17].

A promising opportunity for invisible assessment is using log data collected from educational games. Unfortunately, engineering a system that parses logs is costly and time-consuming. For example, prior work [18, 19] has relied extensively on subject matter expertise to define a student model in form of a Bayesian Network to build both the competency model, and to extract features of the performance data.

Our motivation is that invisible assessment from game data may become more accurate and cheaper to implement if the domain knowledge engineering could be automated by a data-driven process. Game data is often logged with what we call a *slot and filler* structure. In Table 1, we show a simplified example of a real educational game log that uses slot and filler structure. The slots are discrete sets of events that are initiated by the leaner. Each slot may accept zero to multiple fillers. Each filler represents a value of a property of the slot event. For example, a `Move Object` event that represents the learner moved an object in the screen, may have an *x* and *y* coordinates as fillers to represent the target position in two-dimensional space.

Conventional machine learning algorithms cannot input slot and filler data, as they work in structures called *feature vectors* or *sequences*. Others [4] have compared feature vectors and sequences in educational applications. A feature vector representation requires mapping an observation onto a fixed number of features. It is not obvious how to best map sequences of student actions that can be of an arbitrary length into a feature vector that needs to be of a predetermined dimension. Traditional feature vector classifiers (like logistic regression or decision trees) that are used in off-the-shelf data mining packages, such as Weka [5], cannot use slot and filler data as input. In contrast, machine learning algorithms that allow sequential

Id	User Id	Event Name	Event Data
1	ABC	Game Start	{ }
2	ABC	Move Object	{X:363,Y:82}
3	ABC	Move Object	{X:361,Y:54}
4	ABC	Open Toolbox	{}
4	ABC	Activate Tool	{tool:gluumi}
5	ABC	Use Gluumi	{sizeGluedTo:8,sizeNew:9}
...			

Table 1: An example fragment of a log from an educational game.

representations, like Hidden Markov Models (HMM), require a parametric model with the same number of dimensions for all of the observations. This does not occur in slot and filler structures. For example, in Table 1, the `Move Object` slot requires `x` and `y` fillers, while the `Use Gluumi` slot requires only `size` fillers.

In this paper, we propose *Student PRofficiency INferrer from Game data* (SPRING), a novel data analysis pipeline that models game playing behavior. SPRING allows modeling raw data in slot and filler structure. We demonstrate our approach on real student game playing data. Our experiments suggest that SPRING can be used to predict student proficiency without costly domain expertise.

2. SPRING ALGORITHM

SPRING is designed in a way that can capture sequential decision making process of students in a way that is representative of their mastery with minimum reliance on expert knowledge. Our data analysis pipeline receives raw data of student interactions with the different levels of educational game in slot and filler format along with their post-test results and creates a regression model for predicts the post-test score. Algorithm 1 describes the SPRING algorithm. Our formulation allows to aggregate evidence collected from multiple games. The two main steps of our pipeline are learning a sequential and feature vectors models.To learn the sequence model, we first transform the slot and filler observations into discrete (multinomial) indicators.Next, we cluster students according to their performance on an assessment, for example into high or low performing groups. We then learn a HMM per game and cluster. For the regression step, we build a regression model that predicts the student grade by automatically extracting features from the HMMs. We now explain the different steps in detail.

2.1 Discretization

We now describe how we discretize the slot and fillers, (§ 2.1.1) and the student performance (§ 2.1.2).

2.1.1 Slot-Filler Discretization

This step of the data analysis pipeline receives time-ordered student interactions in slot and filler format as input. We give an example of how to transform the slot and filler observations into a discrete (multinomial) observations appropriate for sequence modeling. Consider the `Move Object` slot from Table 1. We wish to transform the different possible fillers into discrete units. Figure 1a shows a screenshot of a minigame

Procedure 1 The SPRING algorithm

Input: : A log file $\mathbf{L_{g,s}}$ of slot-and-filler structure for each game g and student s, student assessment y_s for each student

for each game g **do**
  for each slot **do**
    $\mathbf{D'_{g,s}} \leftarrow$ Discretize_Fillers($\mathbf{L_{g,s}}$)
    $\mathbf{D_{g,s}} \leftarrow$ Predict_Fillers($\mathbf{L_{g,s}}$))
  end for
  $\langle s,c \rangle \leftarrow$ Cluster_Students(y_s)
  for each cluster c **do**
    $\mathbf{D'} \leftarrow \mathbf{D_{g,s}}$ where $s \in \langle s,c \rangle$
    $\theta_{g,c} \leftarrow$ Learn_HMM($\mathbf{D'}$)
  end for
end for
for each sequence d in $\mathbf{D_{g,s}}$ **do**
  $\phi_{g,s} \leftarrow$ Extract_Features($d, \theta_{h,g}, \theta_{l,g}$)
end for
Learn a model that predicts y_s from $\phi_{g,s}$

Discretization

Sequence model

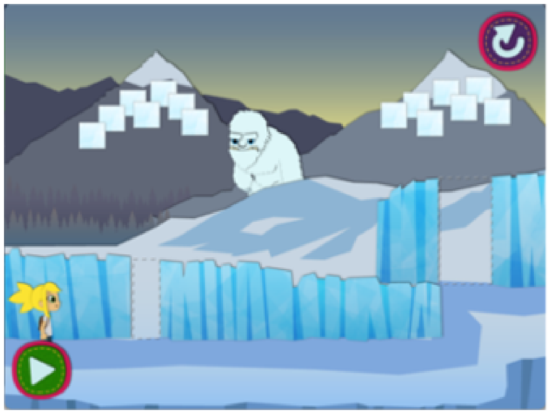
Feature vector model

that generates `Move Object` slots. The purpose of this game is for learners to move ice cubes from the mountain top onto the two designated areas to prevent the Yeti from crossing the wall. In Figure 1b shows a scatterplot of the x and y fillers aggregating across 50? students.

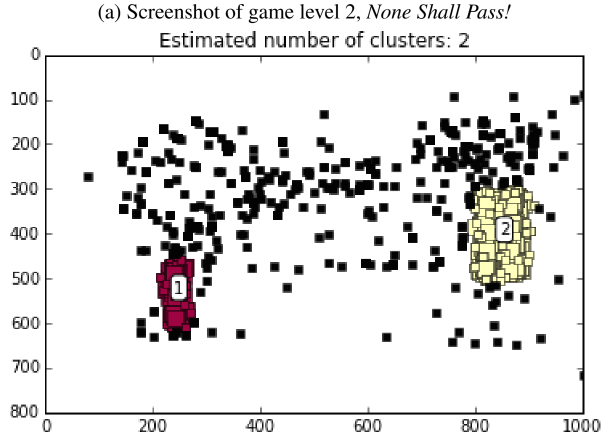
In order to transform the slot and filler observations into discrete events, we use a clustering algorithm for the fillers of each slot. For example, in Figure 1b,we have three clusters, cluster one and two (red and yellow respectively) that represent frequent movements, and cluster zero (black) that represents "outlier" movements. We hypothesize that the outlier group is a result of either technical glitches in the gaming environment or misconceptions. To build discrete events we use the concatenation of slot and cluster labels as a observations in the sequence modeling phase. For example, instead of modeling from an infinite domain like "`Move Object` $\langle x:363, y:83 \rangle$", we use a number of discrete observations like "`Move Object Cluster 1`".

The specific clustering algorithm we used for learning a SPRING model is DBSCAN [2]. In preliminary experiments we tried other unsupervised and non-parametric methods, but we found DBSCAN easier to use because it can find arbitrarily shaped clusters and does not require one to specify the number of clusters in the data a-priori.

Clustering algorithms, like DBSCAN, often can only discretize observations in the training dataset, and would not generalize to unseen fillers. To work around this limitation, we build the capability of transforming any fillers into a cluster label. A K-Nearest neighbor (K-NN) classifier is a good candidate to do this, because both DBSCAN and KNN it also uses euclidean distance. For this, we learn to predict a cluster label for any filler. The details of the discretization phase is demonstrated in Procedure 2. Show an example input (slot-filler), and an example output (discrete seq



(a) Screenshot of game level 2, *None Shall Pass!*



(b) Clusters found using DBSCAN method. We transform each movement action into corresponding cluster id during the discretization step.

Figure 1: Analysis of the `Move Object` slots and fillers from the *None Shall Pass!* mini game

2.1.2 Student performance clustering

Describe this, please

2.2 Sequence Modeling

Hidden Markov Models are good a candidate for the task of unsupervised analysis of sequential data. The use of hidden Markov models for clustering sequences appears to have first been mentioned in [8] and subsequently used in the context of discovering subfamilies of protein sequences [9]. However, it is important to note that the motivation for this problem comes from the goal of learning a *descriptive* model from the data, rather than a *predictive* model. Given the sequence of student actions, we aim to infer meaningful sequence of (latent) states, which describe the process that generated the actions, along with statistical patterns that can describe and distinguish those states.

Learning the "best" value for number of clusters, K, is a difficult problem in practice even for non-dynamic models. There has been considerable prior work on this problem that used penalized likelihoods [16], Monte-Carlo cross-validation [20], and mixture of HMMs [21]. However, for simplicity purposes,

Procedure 2 The Discretization Step of SPRING

Input: : S_g, Sequence of slot-and-filler observations in game level g

procedure DISCRETIZE(S)
    A = empty dictionary of slots and possible fillers
    for each sequence s in S **do**
        for each action a in s **do**
            if a.filler $\neq \varnothing$ **then**
                A[a.slot].append(a.filler)
            end if
        end for
    end for

    for each slot and fillers tuple in A **do**
        clustersIDs = DBSCAN(a.fillers)
        C_s = K-NN classifier on fillers and clusterIDs
    end for

    D = [][] *–2D array of multinomial observations*
    for each sequence *seq* in S **do**
        for each slot and filler tuple $\langle s, f \rangle$ in *seq* **do**
            $D_{i,j} = C_s$.predict(f)
        end for
    end for
    return D
end procedure

we hypothesized that high- and low-performing students have similar usage patterns that are representative of their sequential decision-making process. Instead of learning the number of clusters from data, we used the post-test scores to divide students into two groups and learned a HMM for each group.

We used the Hierarchical Diriclet Process HMM (HDP-HMM) [3], which is allows state spaces of unknown size to be learned from data. HDP-HMM defines a *hierarchical Dirichlet process* prior on transition matrices over countably infinite state spaces and is able to make a principled choice of how many states it needs based on the complexity of its training data. For details on training methods please refer to [3].

We used the output of the discretization step (2D array of multinomial observations) and trained two HDP-HMMs, one for high-performing and the other for low-performing students in each game level. The two models can be considered as a stochastic representation of the sequence of actions and we can use them to infer the likelihood of any arbitrary sequence as a feature for the regression step.

2.3 Regression

For each student s in each game level g, we calculate the difference ($d_{s,g}$) of the likelihood of belonging to high performing group minus the likelihood of of belonging to the low performing group. We calculate this likelihood by estimating the Forward-Backward probabilities on each student sequence sequence of actions based on the two HMMs parameter we estimated (whether the student is in the high performing group

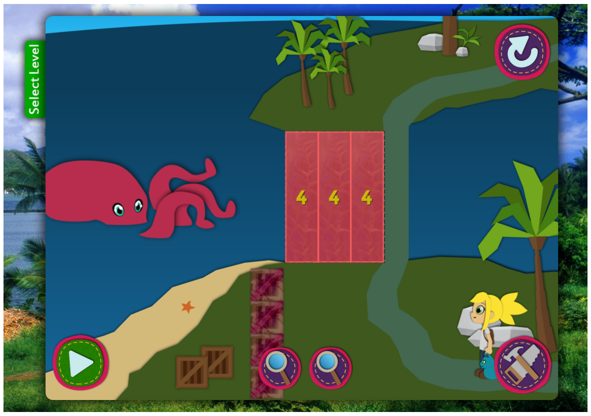


Figure 2: A screenshot of hint provided in game level 11, *You Kraken Me Up!*, in *Alice in AreaLand*. Students should combine four squares into a column and create three copies of the column to cover the designated area and prevent the octopus from attacking *Alice* while she crosses the bridge.

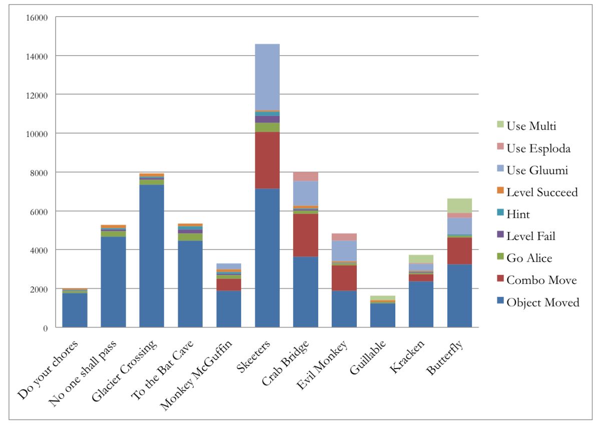


Figure 3: Frequency of Events in Each Game Level

or in the low performing group):

$$d_{s,g} = \theta_{g,\text{high}} - \theta_{g,\text{low}} \tag{1}$$

We use a linear regression model for predicting the post-test scores:

$$\hat{y_s}(\beta) = \sum_g \beta_g \cdot d_{s,g} + \beta_0 \tag{2}$$

Here, β_0 is just an intercept for the model. We optimized the parameters of the model using a 5-fold cross validation on our development set. We experimented with different regularization methods, but only report LASSO [23] as it worked best in our preliminary comparisons:

$$\beta^* = \underset{\beta}{\operatorname{argmin}} ||y_s - \hat{y_s}(\beta)||_2 + \lambda \cdot ||\beta||_1 \tag{3}$$

3. EMPIRICAL EVALUATION

3.1 Game Environment

Alice in AreaLand is an educational game developed for research purposes. It focuses on teaching and assessing geometric measurement, specifically the understanding of area, among 6th grade students. The game targets three main stages in the development of area: 1) area unit iteration, 2) use of unit squares to measure area, and 3) use of composites to measure area. The current version has 11 game levels. A simple student scenario involves covering a 2D area with smaller unit squares placed end-to-end in non-overlapping fashion, combining the single squares into rows or columns, and then determining the number of rows or columns needed. Figure 2 shows a screenshot of one game level.

Throughout the game, *Alice* is accompanied by *Flat Cat* – an assistant character who provides feedback and scaffolding to the player in the beginning of each game level and upon request when students push a hint button (represented by two magnifiers at the bottom center of Figure 2). Earlier game levels are designed for students to learn about area unit iteration and usually require them to cover a number of predefined areas with unit squares (not necessarily in a non-overlapping fashion). By advancing through game levels, students are presented with three tools: *Gluumi* for combining unit squares by gluing them together; *Multy*, for making copies of different objects; and *Esploda* for breaking compound shapes into single units. There is no limit for completing a game level regarding time or number of actions students may execute. The students press the *Go Alice* button (bottom left corner of Figure 2) if they deem their performance to be satisfactory for *Alice* to proceed. Based on the covered area and the arrangement of the tiles, they either advance to the next level or receive a feedback and stay in the same level.

3.2 Dataset

Our dataset consists of time-stamped interactions of 129 students in 11 game levels. For 77 students, we also have post-test scores from a paper-based exam with 20 questions in the 3 skills of geometric measurement. In total, there are 88,458 events recorded in the dataset from 1,510 game sessions, meaning that student tried some of the game levels for multiple times. Based on the ECD framework, beginning levels only involve area unit iteration skill and the other skills and related features are gradually added to the later game levels. Figure 3 shows the frequency of different events in each game level. As depicted in Figure 3, the student interactions with the system in all game levels is dominated by movements.

We only used the interactions of the students who participated in the post-test. In the case of multiple attempts in a game level, we only considered the interactions from the first attempt. Figure 4 shows the boxplot of sequence length in each game level.

3.3 Experimental Setup

First we divide students into two groups, one (%80) for training and development purposes and the other (%20) for test and verification. In the synthesis phase, we transform log data of different nature (eg. movements, use of the tools, requests for

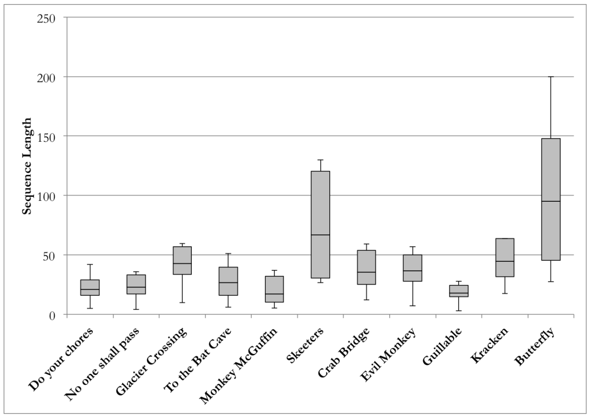


Figure 4: Boxplot of Sequence Length in Each Game Level

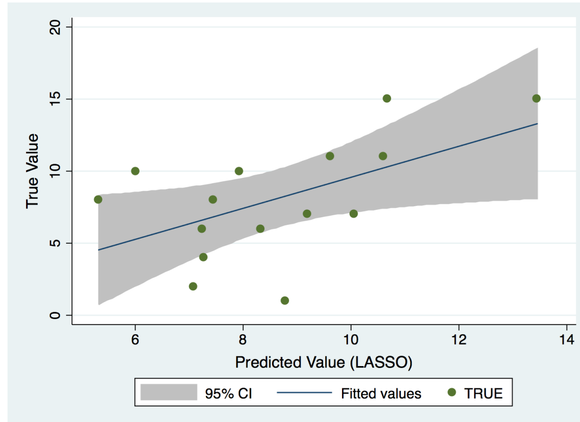


Figure 5: Results of predicting the post-test scores of the held-out set along with the regression line and %95 interval.

hints) in each game level to observable values that can be used as evidence for learning. In the modeling phase, we use the observations to train two Hidden Markov Model (HMM)s that capture the sequence of actions for high- and low-performing students. In the aggregation phase, we uses the likelihood of students' sequence of action in order to build a regression model that is predictive of the post-test results. Finally, we test the regression model on the held-out set (%20) and report the results.

3.4 Results

In order to evaluate our regression model we decided to compare its performance agains two baselines: 1) A regression model that uses success and failure of students in each game level as feature and 2) A regression model that uses the normalized sequence lengths in each game level. Our model was able to significantly outperform both baselines regarding to mean absolute error and root mean squared error. Moreover, the R^2 correlation and Spearman ρ of our predicted values with the true values was much higher than both baselines. Table 2 shows the results.

Predictive Features	R^2	ρ	**MAE**	**RMSE**
Sequence Length (Normal)	0.06	0.79	3.24	4.15
Success / Failure	0.01	0.63	3.22	4.14
SPRING	0.55	0.82	2.84	3.35

Table 2: The results of predicting post-test scores using three different feature sets

Figure 5 shows the true values vs. predicted values using LASSO regression along with the regression line and %95 confidence interval.

The regression model also provides us with an insight into how important each game level is in distinguishing between high- and low-performing students. Table 3 shows the weights of each game level in the best perfoming model.

As it is shown in the table, game level 11 (*You Kraken Me Up!* – screenshot Figure 2), has the highest weight in the regression model. This can be interpreted as the difference between the sequence of student interactions in this game level has the highest factor in distinguishing between high- and low-performing students.

3.5 Discussion

In this study we aimed to explore the effectiveness of data-driven methods for student modeling in educational games. To demonstrate the effectiveness of our framework, we used features from it to predict the post-test results using a regression model and showed significant improvement over two baselines based on intuition. However, predicting post-test scores was not the only reason for designing the pipeline. We were also seeking for a descriptive model that can give us insight into low-level patterns in student movements. This was the main idea behind using HDP-HMMs in the sequence modeling phase and our hypothesis that high-performing and low-performing sudents have different sequential patterns.

Figure 6 shows the difference between two HMMs learned for one of the game levels. In this game level in the initial state, students are presented with composite objects and they have to use the `Esploda` tool to convert them into single unit objects. Then, they have to use the `Gluumi` tools to convert the single unit objects into composite objects that fit the designated area. And finally, they have to move the composite object they have created into the designated area. For comparison purposes, we are showing a three state HMM for both high-

Game Level	**Weight**
You Kraken Me Up!	1.686
Skeeterz	1.059
Evil Monkeys	1.053
To the Bat Cave	0.855
None Shall Pass	0.417
...	

Table 3: Weights of each game level in the best performing model.

and low-performing students and removed the edges that had the probability below 0.05. Figure 6a represents the HMM for high-performing students. The three latent states, S_{0-2} can be interpreted as sequential steps, that students took for solving this problem. As illustrated in Figure 6a, high-performing students follow the expected path of using the `Esploda` tool in S_0, using the `Gluumi` tool in S_1, and then moving the composite objects into the designated areas `Move Combo_ClusterID` untill they successfully finish the game. On the other hand, low-performing students (Figure 6b) do not necessarily follow the expected path and the probability of moving object into the outlier clusters (`Move Object_0 and Move Combo_0`) is also higher in each state for low-performing students.

4. RELATION TO PRIOR WORK

In order to deal with such highly unstructured data, researchers often use carefully designed network structures (such as Bayesian Networks [1, 18]) or game-specific heuristics and benchmarks generated by experts playing the game [14, 22]. However, this approach is extremely labor intensive and might fail to capture meaningful patterns in student exploratory habits within the game. Given these limitations, data-driven analysis of student interactions provides a powerful alternative that facilitates the discussions around what does and does not work in a particular educational game.

The potential of computer games for educational purposes has been of interest since nearly the beginning of videogames. Unlike video games, which focus on creating an entertaining experience for the user, educational games require principles and strategies that engage students while maximizing their learning gain. Therefore, data-driven analysis of student behavior is crucial to better understand the learning process and improve the tools in the future.

There have been numerous attempts among the educational research community to develop analytic methods and build predictive models based on the data from educational games. *Newton's Playground* is an ECD based educational game with 74 problems that designed to teach qualitative physics to students in eighth- and ninth-grade. Students have to guide a green ball to a red ball by creating simple machines. Everything obeys the basic rules of physics relating to gravity and Newtons three laws of motion. Shute et al. [18] studied the effect of ECD design on student learning and found that students who played the game in a 4 hour session, showed significant improved in their qualitative, conceptual physics understanding.

Rumble Blocks is another educational game designed to teach basic concepts of structural stability and balance to children in grades K-3 (ages 5-8 years old). Harpstead et al. [6], studied the alignment of game to its target learning goals by examining whether student solutions follows the targeted principals. They employ clustering techniques on the individual solutions created by actual students and use principle-relevant metric (PRM) to measure how closely the representative solution embodies a specific targeted principle. The results demonstrated a misalignment between the feedback provided to students and the targeted knowledge.

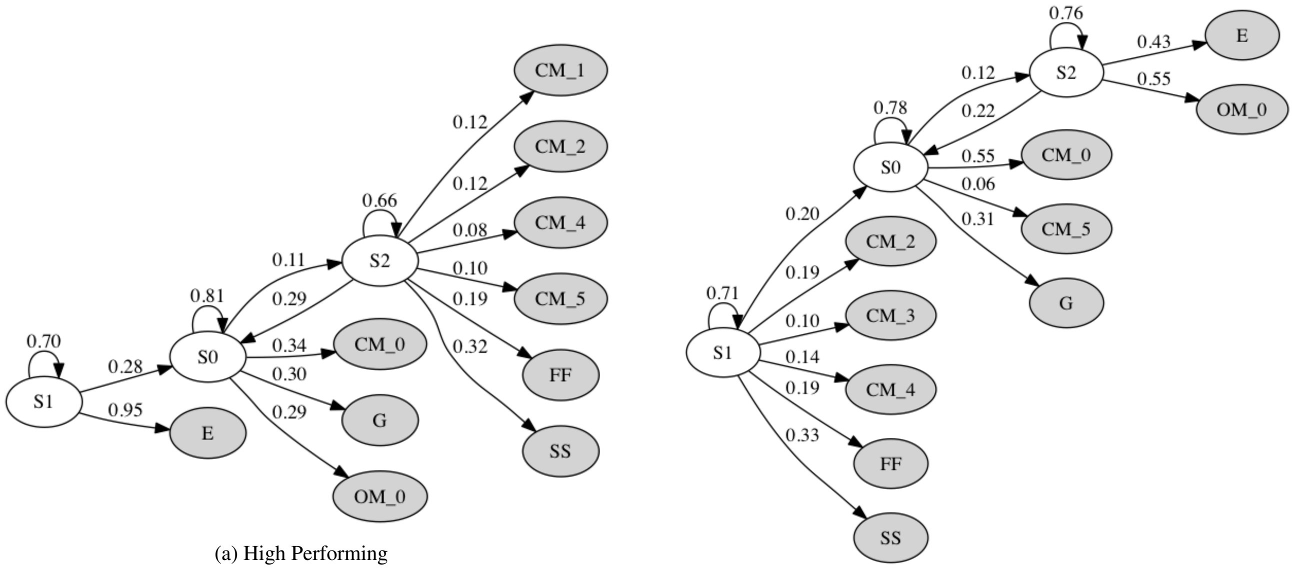
Battleship Numberline is another educational game for understanding fraction using number line estimation. Students attempt to explode target ships and submarines by estimating numbers on a number line. Lomas et al. [13] performed a large-scale online experiment in order to study the effect of challenge on player motivation and learning. They presented different configurations of the game for different groups of students and used a combination of time spent and challenges attempted as a measure of engagement and the average success rate of each design configuration as a measure of challenge. The results showed a linear correlation between challenge (difficulty) and engagement, meaning the easier the game, the longer students played.

Refraction is another educational game for learning about fractions by splitting laser beams into fractional amounts to target spaceships by avoiding asteroids. Liu et al. [12], created an ensemble algorithm that combines elements of Markov models, player heuristic search, and collaborative filtering techniques with state-space clustering in order to predict player movements on last game-level based on the history of movements in previous game levels. Lee et al. [11] extended the former framework by building state-action graph and using feature selection techniques to reduce the number of features for each state. To ensure extensibility, they also tested the framework on another game *DragonBox* and reported improvement over a Markov predictor.

5. CONCLUSION

Modeling student behavior in open-ended environments such as educational games is an interesting and complex problem. Particulary data-driven models, can be valuable tools for game designers, educators, and researchers for analyzing how students learn different skills by using their systems. In this paper, we presented a data analysis framework that is able to learn a model from game activity logs that is predictive of student proficiency with minimum reliance on expert knowledge about the game environment. One of the key drawbacks of model-free methods is that their results are difficult to interpret. However, the discretization process along with the use of HDP-HMM, makes our model human readable and can give us insight into how student interact with each game level. Moreover, the parameters of the regression model provides a good intuition about the importance of each game level and its influence in distinguishing between high- and low-performing students.

There are many possible directions for future work. On a lower level we can integrate the discretization and sequence modeling step by designing a new type of Hidden Markov Model that accepts slot and filler sequences as input (work in progress). On a higher level, instead of dividing students into two groups (high- and low-performing), we can use the new type of HMM in order to cluster students into more groups that might be more representative of different approaches students follow in order to solve each game level. We can also replace our regression model with a more powerfull ensemble method that considers different subskills in order to integrate features of each game levels in order to predict the post-test scores.



(a) High Performing

(b) Low Performing

Figure 6: Two Hidden Markov Models learned from data in one of the game levels. The left HMM represents the high-performing students and the right one represents the low-performing ones.

We only used data from *Alice in AreaLand* in order to evaluate our model. However, the model is designed in a way that can model sequence of student actions in other similar gaming environments that use slot-and-filler structures in order to log student interactions with the system. Finally, using our framework we will be able to build a model that detects incorrect strategies or common misconception that can be used in a dynamic hinting system in order to increase player engagement and learning.

Limitation, we don't compare to Valerie Schultz

Limitation, we don't control for game ability like Valrerie does

Limitation, we only used data from Alice in wonderland

Method is accuaate

6. REFERENCES

1. David W Albrecht, Ingrid Zukerman, and An E Nicholson. 1998. Bayesian models for keyhole plan recognition in an adventure game. *User modeling and user-adapted interaction* 8, 1-2 (1998), 5–47.
2. Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
3. Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. 2008. An HDP-HMM for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*. ACM, 312–319.
4. José P. González-Brenes and Jack Mostow. 2011. How to Classify Tutorial Dialogue? Comparing Feature Vectors vs. Sequences. In *Proceedings of the 4th International Conference on Educational Data Mining* (Eindhoven, Netherlands), M. Pechenizkiy and T. Calders and C. Conati and S. Ventura and C. Romero and J. Stamper (Ed.). 169–178. http://educationaldatamining.org/EDM2011/wp-content/uploads/proc/edm2011_paper23_full_Gonzalez-Brenes.pdf
5. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.
6. Erik Harpstead, Christopher J MacLellan, Vincent Aleven, and Brad A Myers. 2014. Using extracted features to inform alignment-driven design ideas in an educational game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3329–3338.
7. Peter Hofman, Bryan Goodwin, and Stuart Kahl. Re-Balancing Assessment: Placing Formative and Performance Assessment at the Heart of Learning and Accountability. (????).
8. Biing-Hwang Fred Juang and Lawrence R Rabiner. 1985. A probabilistic distance measure for hidden Markov models. *AT&T technical journal* 64, 2 (1985), 391–408.
9. Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of molecular biology* 235, 5 (1994), 1501–1531.

10. M Lazarín. 2014. Testing overload in Americas schools. *Center for American Progress. Retrieved from www.americanprogress.org/issues/education/report/2014/10/16/99073/testing-overload-in-americas-schools* (2014).

11. Seong Jae Lee, Yun-En Liu, and Zoran Popovic. 2014. Learning Individual Behavior in an Educational Game: A Data-Driven Approach. In *Educational Data Mining 2014*.

12. Yun-En Liu, Travis Mandel, Eric Butler, Erik Andersen, Eleanor O'Rourke, Emma Brunskill, and Zoran Popovic. 2013. Predicting player moves in an educational game: A hybrid approach. In *Educational Data Mining 2013*.

13. Derek Lomas, Kishan Patel, Jodi L Forlizzi, and Kenneth R Koedinger. 2013. Optimizing challenge in an educational game using large-scale design experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 89–98.

14. Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. 2014. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1077–1084.

15. Robert J Mislevy, John T Behrens, Kristen E Dicerbo, and Roy Levy. 2012. Design and discovery in educational assessment: evidence-centered design, psychometrics, and educational data mining. *JEDM-Journal of Educational Data Mining* 4, 1 (2012), 11–48.

16. Lawrence R Rabiner, CH Lee, BH Juang, and JG Wilpon. 1989. HMM clustering for connected word recognition. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*. IEEE, 405–408.

17. Valerie J Shute, Eric G Hansen, and Russell G Almond. 2008. You can't fatten a hog by weighing it-Or can you? Evaluating an assessment for learning system called ACED. *International Journal of Artificial Intelligence in Education* 18, 4 (2008), 289.

18. Valerie Jean Shute and Matthew Ventura. 2013. *Stealth assessment: Measuring and supporting learning in video games*. MIT Press.

19. Valerie J Shute, Matthew Ventura, Malcolm Bauer, and Diego Zapata-Rivera. 2009. Melding the power of serious games and embedded assessment to monitor and foster learning. *Serious games: Mechanisms and effects* 2 (2009), 295–321.

20. Padhraic Smyth. 1996. Clustering Using Monte Carlo Cross-Validation.. In *KDD*. 126–133.

21. Padhraic Smyth and others. 1997. Clustering sequences with hidden Markov models. *Advances in neural information processing systems* (1997), 648–654.

22. Bulent Tastan and Gita Reese Sukthankar. 2011. Learning Policies for First Person Shooter Games Using Inverse Reinforcement Learning.. In *AIIDE*.

23. Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.