RESEARCH-ARTICLE

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

MOHAMMAD HASSAN FALAKMASIR, University of Pittsburgh, Pittsburgh, PA, United States

JOSÉ P GONZALEZ-BRENES

GEOFFREY J GORDON, Carnegie Mellon University, Pittsburgh, PA, United States

KRISTEN E DICERBO, Pearson plc, London, U.K.

Open Access Support provided by:

University of Pittsburgh

Carnegie Mellon University

Pearson plc

Published: 25 April 2016

Citation in BibTeX format

L@S 2016: Third (2016) ACM Conference on Learning @ Scale
April 25 - 26, 2016
Scotland, Edinburgh, UK

L@S '16: Proceedings of the Third (2016) ACM Conference on Learning @ Scale (April 2016)
https://doi.org/10.1145/2876034.2876038
ISBN: 9781450337267

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

Student assessments are important because they allow collecting evidence about learning. However, time spent on evaluating students may be otherwise used for instructional activities. Computer-based learning platforms provide the opportunity for unobtrusively gathering students' digital learning footprints. This data can be used to track learning progress and make inference about student competencies. We present a novel data analysis pipeline, Student Proficiency 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. Moreover, it produces a simple interpretable model that not only fits the data but also predicts learning outcomes. We validate our framework using data collected from students playing 11 educational mini-games. Our results suggest that SPRING can predict math assessments accurately on withheld test data (Correlation=0.55, Spearman ρ=0.51).

Author Keywords

Educational Games, Student Modeling, Stealth Assessment

1. INTRODUCTION

Educational assessments are important because they collect evidence about whether instructional goals are achieved or not. Unfortunately, the process of administering assessments is usually disconnected from the instructional environment, and it is often disruptive to the learning experience. In many developed countries, students now find themselves spending increasing amounts of time preparing for and taking tests instead of learning [11]. For example, a survey of the current state of testing in the United States of America revealed that students are taking an average of 113 standardized tests between pre-K and high school [12]. For these reasons, it is not surprising that both politicians and the general population have been weighing in on the question of whether students are being tested too much [12].

Learning and succeeding are not easily measured by solving constrained tasks and answering verbal or multiple-choice questions. Advancements in educational measurements allow us to diagnose student competencies more accurately and effectively at various levels during the course of learning. According to Evidence Centered Design (ECD) [16], the goal of assessment is to characterize the evidence regarding claims one wants to make about the competencies of individuals or groups of students. The assessment process involves identifying, organizing, or creating activities for students so that we may observe that evidence. An alternative to traditional summative assessment (e.g, exams) is to use data collected from computer-based learning environments. This data can be used for invisible (or stealth) assessment [22]. One of the advantages of invisible assessment is to blur the distinction between assessment and learning.

Analyzing log data collected from students playing educational games is a promising way to perform invisible assessment. Studying the sequence of actions in an educational game provides incremental evidence about mastery of specific skills and allows us to infer what students know at any point in time. Prior studies have also reported that invisible assessment may reduce test anxiety while not sacrificing validity and reliability of the results [21]. Unfortunately, engineering a system that interprets log data is costly and time-consuming. For example, Shute et al. [22, 23] relied extensively on domain expertise to define a *Competency Model* in the form of a Bayesian Network to describe the set of knowledge and skills involved in the learning environment. They also conducted feature engineering on the performance data to build an *Evidence Model* that expresses how student interactions with the system constitute evidence about competency model variables.

In this paper, we propose *Student PRoficiency INferrer from Game data* (SPRING), a novel data analysis pipeline that models game playing behavior. 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. Our experiments suggest that SPRING is able to capture student differences and provide features that can be used to predict student proficiency. The rest of this paper is organized as follows: § 2 describes the specific challenges that make game data difficult to analyze; § 3 details the SPRING algorithm; § 4 evaluates the SPRING algorithm with student data; § 5 relates to prior work; § 6 provides concluding remarks.

L@S 2016, April 25–26, 2016, Edinburgh, UK.
Copyright © 2016 ACM 978-1-4503-3726-7/16/04 ...$15.00.
DOI: http://dx.doi.org/10.1145/2876034.2876038

2. SLOT AND FILLER DATA

Game data is often logged in a format that we call a *slot and filler*. In Table 1, we show a simplified example of a real educational game log that uses slot and filler structure. The Event Name column lists a number action or events that have been logged. For each event, there is a set of slots (properties) that further describe the event. Each filler represents a property value of the slot. In the context of this paper, we assume that each slot takes exactly one filler. For example, a `Move Object` event that represents the learner moving an object on the screen may have x and y coordinates as slots and the target position in two-dimensional space $(361, 82)$ as the fillers.

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	{}
5	ABC	Activate Tool	{tool:gluumi}
6	ABC	Use Gluumi	{sizeGluedTo:8,sizeNew:9}
...			

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

Analyzing slot and filler data is difficult. Some general-purpose predictive models (like logistic regression or decision trees) that are used in off-the-shelf data mining packages, such as Weka [8], use a vector representation of features as input. A vector representation requires mapping of observations onto a fixed number of features. It is not obvious how to best map the sequence of events (with slot and filler modifiers) into feature vectors. Feature vectors need to be of a predetermined dimension, and a game log may have a variable number of events and slots.

Machine learning algorithms that are able to work with data that is not represented as feature vectors include kernel methods and graphical models. Popular general-purpose graphical models, such as Hidden Markov Models (HMM), allow structure learning from sequential data with a variable number of events. However, these models usually require input observations with the same number of dimensions for each time step. It is not clear how to transform observations when the number of slots can vary for different events. For example, in Table 1, the `Move Object` requires 2 slots, while the `Game Start` has no slots. On the other hand, *ad hoc* graphical models have been proposed for mining specific games. However, these graphical models were tailored by subject matter experts and do not generalize across different games [20].

SPRING models slot and filler data for educational games when domain knowledge is too expensive or cumbersome to collect. To learn a SPRING model, we need (i) logs from students playing educational games, (ii) and their results in traditional assessments. SPRING transforms the slot and filler data to sequences that can be modeled with Hidden Markov Models. It then extracts features from the sequences and uses them in a regression model to predict the score of the traditional assessment. After this, SPRING is able to predict assessment performance of students by just observing their game activity as described by their slot and filler log.

3. SPRING ALGORITHM

SPRING is designed to capture the 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 in slot and filler format and extracts features to build a model that predicts post-test scores from game playing data. SPRING assumes that an event is always described with the same set of slots. For example, an event such as `Move Object` always would be described with x and y slots.

Procedure 1 The SPRING algorithm

```
Input: : A log file $\mathbf{L_{g,s,t}}$ of events (with slot and fillers) for
    each game $g$ and student $s$ (indexed by time $t$), results
    from a student assessment $y_s$, number of performance
    clusters $K$, number of latent states $H$.
for each game $g$ do                                              } Discretization
    $\mathbf{D_{g,s,t}} \leftarrow$ Discretize_Fillers($\mathbf{L_{g,s,t}}$)
    $\langle s,c \rangle \leftarrow$ Cluster_Students($y_s, K$)
    for each student performance group $c$ do                     } Sequence Model
        $\mathbf{D_c} \leftarrow \mathbf{D_{g,s,t}}$ where $s \in \langle s,c \rangle$
        $\theta_{g,c} \leftarrow$ Learn_HMM($\mathbf{D_c}, H$)
for each sequence $d$ in $\mathbf{D_{g,s}}$ do                     } Feature Vector Model
    $\phi_{g,s} \leftarrow$ Extract_Features($d, \theta_{g,c}$)
Learn a model that predicts $y_s$ from $\phi_{g,s}$
```

Procedure 1 describes the three main steps of our pipeline: discretization, sequence modeling, and feature vector modeling. The discretization step transforms filler observations into multinomial indicators that are used as evidence for learning. The sequence modeling step learns a representation from data in the form of a Hidden Markov Model for high- and low-performing students. The feature vector modeling step uses the likelihood of student sequences in each game level as a feature to predict the post-test results. We now explain the different steps in detail.

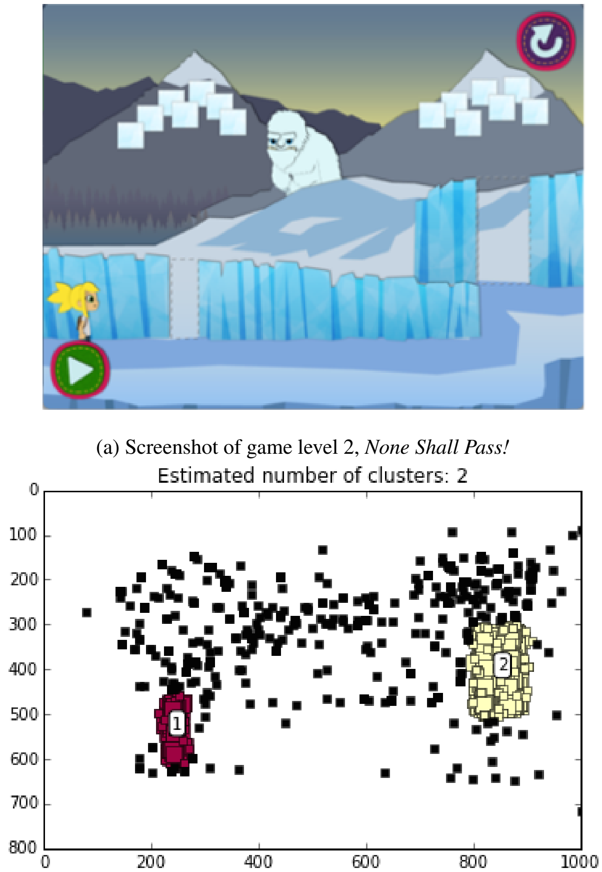
3.1 Discretization

We now describe how we discretize the slot and fillers (§ 3.1.1) and the student performance (§ 3.1.2).

3.1.1 Slot-Filler Discretization

Consider the `Move Object` event from Table 1. We wish to transform the different possible fillers into discrete units. Figure 1a shows a screenshot of a minigame that generates `Move Object` events. 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. Figure 1b shows a scatterplot of the x and y fillers aggregating across 77 students.

Procedure 2 describes the method we use to discretize fillers for each game. The input is time-ordered student interactions



(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.

in slot-and-filler format for a single game g. The purpose is to transform the fillers of an event into discrete (multinomial) observations appropriate for sequence modeling. For this, we use an unsupervised clustering algorithm for the fillers of each event. For example, in Figure 1b, we show three clusters: cluster one and two (red and yellow respectively) that represent frequent movements, and cluster zero (black) that represents "outlier" movements. We speculate that the outlier group is a result of either technical glitches in the gaming environment or student misconceptions. To build discrete events we use the concatenation of event and cluster labels as an observation 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 in our experiments is DBSCAN [6]. In preliminary experiments, we tried other unsupervised 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 advance. The time complexity of DBSCAN is $O(n \log n)$ (n = the number of data points) and there are recent parallelized implementations that are scalable [4, 17].

Procedure 2 The filler discretization step of SPRING

```
Input: : $L_{g,s,t}$, sequence of slot-and-filler observations in game level $g$
procedure DISCRETIZE_FILLERS($\mathbf{L_{g,s,t}}$)
    $\mathbf{Z} \leftarrow$ enumerate_slots($\mathbf{L_{g,s,t}}$)
    for each event $z$ do
        $\mathbf{F_z} \leftarrow$ get_fillers($\mathbf{L_{g,s,t}}$)
        $\langle \mathbf{F_z}$ , clustersIDs $\rangle \leftarrow$ Cluster($\mathbf{F_z}$)
        model $\leftarrow$ learn_classifier($\mathbf{F_z}$, clustersIDs)

    for each student $s'$ do
        for each time step $t'$ do
            if $z ==$ slot($L_{g,s',t'}$) then
                $\hat{f} \leftarrow$ model.predict(filler($L_{g,s',t'}$))
                $d \leftarrow$ concat(z, $\hat{f}$)
                $\mathbf{D_{g,s',t'}} \leftarrow d$
    return $\mathbf{D}_{g,s,t}$
```

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 trained a classifier for transforming any filler into a cluster label. In particular, we used a K-Nearest Neighbor (K-NN, K=5) classifier because similar to DBSCAN, it uses Euclidean distance as a similarity metric. The classifier is trained to identify a cluster label using the fillers as predictors. The K-NN classifier might not be the best choice for fillers in the outlier cluster. However, our preliminary experiments using cross-validation showed that it has enough discriminative power to distinguish the outlier fillers from other clusters.

3.1.2 Student Performance Clustering

We use assessment data to group students according to their performance. For this, we simply divide students into two groups by comparing their score to the median score of the population: students who score below or equal to the median are assigned to the low-performing group and the students who received a post-test score higher than the median in the high-performing group.

We did not consider other more sophisticated clustering techniques. Due to the small number of students in our dataset, particularly in the later game levels, we only use two clusters and do not tune the number of performance groups. We hypothesize that students in the two performance groups play the game differently. Our sequence modeling phase should be able to capture these differences by learning the interaction patterns of each group in different game levels.

3.2 Sequence Modeling

Once we have transformed the slot-and-filler structure into sequences of discrete observations, we can start looking for sequential patterns among different performance groups in each game level. Hidden Markov Models are a popular statistical tool for analyzing sequential patterns. We can model the sequence of student actions as observations, and infer a set of unobserved (latent) states. In this context, the states describe the process that generated the observations, along with statistical patterns that can describe and distinguish those states.

Learning the "best" value for the number of states (H) is a difficult problem in practice. There has been considerable prior work on this problem that used penalized likelihoods [18], Monte-Carlo cross-validation [24], and mixtures of HMMs [25]. We used the Hierarchical Dirichlet Process HMM (HDP-HMM) [7], which 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 [7].

In summary, 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.

3.3 Feature Vector Modeling

For each of the game levels g, the HMM parameters of the high performing group $\theta_{g,\text{high}}$, or the low-performing group $\theta_{g,\text{low}}$ can be used to determine the likelihood of a sequence. Consider the sequence of discretized observations for a student s. The likelihood of the sequence given the high-performing HMM is:

$$l_{g,s,\text{high}} = p(\text{event}_1, \text{event}_2, \ldots \text{event}_n | \theta_{g,\text{high}}) \qquad (1)$$

This likelihood can be estimated efficiently with an algorithm called the Forward-Backward algorithm. For each student s in each game level g, we calculate the difference ($d_{s,g}$) between the log-likelihood of observing the sequence of actions given the two models. To scale the values between -1 and 1, we use a function called hyperbolic tangent:

$$d_{g,s} = tanh[K * (l_{g,s,\text{high}} - l_{g,s,\text{low}})] \qquad (2)$$

If one HMM fits the data much better than the other, the value of $d_{s,g}$ is closer to -1 or 1. On the other hand, when the likelihoods are similar, the value of is close to zero. The parameter K represents the scaling factor. Larger values for K result in output distribution that is closer to +1 and -1 in the case of clear difference. We experimented with different K values across game levels and picked K=100.

We use the transformed log-likelihood differences in each game level as features of a linear regression model that predicts the post-test scores:

$$\hat{y}_s(\beta) = \sum_g \beta_g \cdot d_{g,s} + \beta_0 \qquad (3)$$

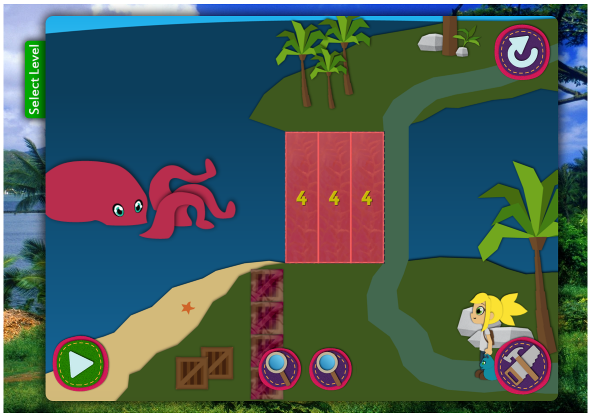


Figure 2: A screenshot of hint provided in game level 11, "You Kraken Me Up!", in Alice in AreaLand. Students should combine four squares into a column and create three copies of the column to cover the designated area and prevent the octopus from attacking Alice while she crosses the bridge.

Here, $\hat{y}_s(\beta)$ is the predicted post-test score, and β_0 is an intercept for the model.

We learn the regression parameters using data:

$$\beta^* = \underset{\beta}{\operatorname{argmin}} ||y_s - \hat{y}_s(\beta)||_2 + \lambda \cdot ||\beta||_1 \qquad (4)$$

Here, y_s is the true score of the student. To avoid over-fitting, we experiment with different regularization methods, but only report LASSO [26] as it worked best in our preliminary comparisons. For our experiments, we optimize the parameters of the model using a 5-fold cross-validation on our development set.

4. EMPIRICAL EVALUATION

We use data collected from an educational game called *Alice in AreaLand* to evaluate SPRING. This game is developed for research purposes as a part of Pearson Insight Learning System[1]. 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 12 game levels. A simple student scenario involves covering a 2D area with smaller unit squares placed end-to-end in a 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 at 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

[1] http://researchnetwork.pearson.com/learning-science/insight-learning-system

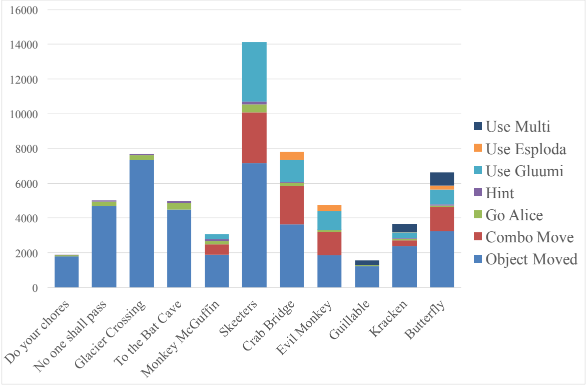


Figure 3: Frequency of Events in Each Game Level

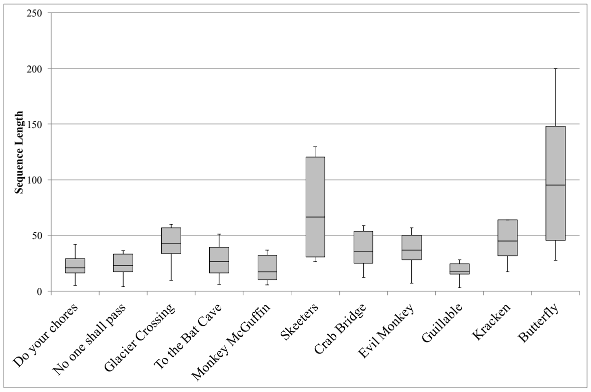


Figure 4: Boxplot of sequence length in each game level

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 feedback and stay at the same level. Beginning levels only involve simple concepts, and they get more complicated while the student progresses.

4.1 Dataset

Our dataset consists of time-stamped interactions of 129 students in 11 game levels. We do not use the data from one of the game levels due to technical issues. For 77 students, we also have post-test scores from a paper-based exam with 20 questions in the 3 skills of geometric measurement as described above. The post-test scores range from 0 to 18 (out of 20) with the mean of 7.81 and standard deviation of 4.36.

In total, there are 88,458 events recorded in the dataset from 1,510 game sessions, meaning that students tried some of the game levels multiple times. Based on the ECD framework, beginning levels only involve the area unit iteration skill and other skills and related tools 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 interaction with the system in all game levels is dominated by movements. Figure 4 shows a boxplot for the sequence length in each game level.

We only use the interactions of students who participated in the post-test (77 students) in our data analysis pipeline. Our dataset includes the records of multiple attempts (sequence of actions) students made to solve each game level. Students may struggle with a game level on the first attempt and change their answers multiple times to find the solution. Different attempts are a rich data source for studying students' change of performance over time. However, we decided to only incorporate the first attempt in each game level in our analysis because SPRING only accepts one sequence of action per game level for each student. Figure 4 shows the boxplot of sequence length in each game level.

4.2 Experimental Setup

We divide students into two groups, one (80%) for training and development purposes, and the other (20%) for test and verification. In the discretization phase, we transform the log data from slot-and-filler structures for each game level into sequences of multinomial variables that can be used as evidence for learning for each student. In the sequence modeling phase, we use the multinomial sequences of students in the development set (80%) to train two HMMs that represent high- and low-performing students. In the feature vector modeling phase, we use the likelihood of students' sequences 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.

4.3 Quantitative Results

We evaluate two implementations of SPRING, each of them using two ($K = 2$) HMMs for each game level as described previously– one for high-performing students and one for low-performing students:

- SPRING ($H = 1$) uses an HMM with only one hidden state, a model with no memory (no temporal information). This model evaluates the effect of our discretization step since it only considers the distribution of the discretized observations in a sequence to distinguish between high- and low-performing students (as in Gaussian Mixture Model).
- SPRING (H^*) learns the best number of states for each game level based on the training examples using a Hierarchical Dirichlet Process [7].

Additionally, we evaluate two baseline methods:

- Success / Failure: A regression model with eleven binary features. Each feature represents whether a student successfully passed each of the 11 different mini-games on the first attempt.
- Sequence Length: A regression model that uses the normalized sequence lengths in each game level.

Table 2 summarizes our results. We use each method to predict the students the assessment score of the students in the test set. We compare the predictions to true scores using popular evaluation metrics: Pearson product-moment correlation coefficient (R), Spearman rank correlation ρ, mean absolute error (MAE) and root mean squared error (RMSE). The best-performing model according to all of the metrics is SPRING H^*.

Predictive Features	R	ρ	**MAE**	**RMSE**
Sequence Length (Normal)	0.15	0.35	2.93	3.61
Success / Failure	0.02	0.16	3.21	4.08
SPRING ($K = 2, H = 1$)	0.17	0.25	3.60	3.10
SPRING ($K = 2, H^*$)	**0.55**	**0.51**	**2.84**	**3.35**

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

The baselines and SPRING have a positive correlation with the true values. However, the correlation is statistically significant only for SPRING H^* (one-sided t-test: $r = 0.55$, $n = 15$, and $p < 0.05$). This suggests that the predicted scores of SPRING H^* are positively correlated with the true assessment scores.

We also calculate the absolute difference between the true and predicted post-test scores for all of the students based on each model. We use a paired one-sided t-test (n=15) to judge the significance of improvement over the baselines. We cannot reject the null hypothesis that the baselines are as good or better than SPRING ($p = 0.20$ over the Success/Failure baseline and $p = 0.26$ over the Sequence Length baseline).

Our results suggest that even though there is a significant and positive correlation between the true and SPRING's predicted post-test scores, the improvement in terms of MAE may not be significant. We notice that changes in correlation are more drastic than changes in MAE. For example, in Table 2 a 0.4 improvement in correlation over the first baseline resulted in only a change of 0.09 and improvement in MAE. It is unclear which evaluation metric is better for our application. A possible explanation of why the MAE is not significant is that our dataset is too small. Our held-out set contains only 15 students and a few of them achieved poor performance results in the post-test while successfully finishing most of the game levels. Future research could investigate how to evaluate invisible assessment systems, as well as replicate our results with larger datasets to see if the observed trend continues.

A limitation of our approach is that our student performance clustering phase uses the post-test scores (independent variable) to divide students into high- and low-performance groups.

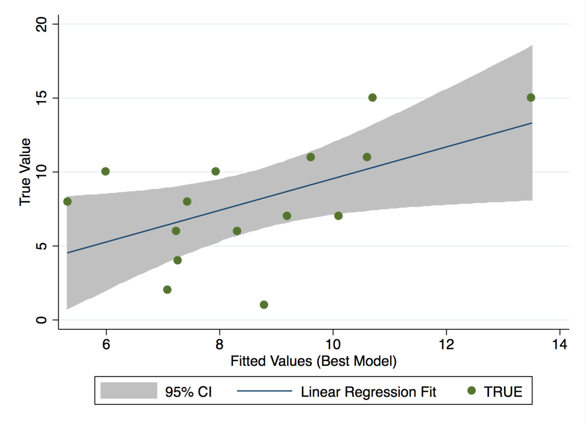


Figure 5: Results of predicting the post-test scores of the held-out set along with the regression line and 95% interval.

In order to avoid using information from the independent variable to calculate the features of our regression model, we are using an 80%-20% split to validate the results instead of for example using a 10-fold cross validation. This design decision, while necessary, reduces the sample size further.

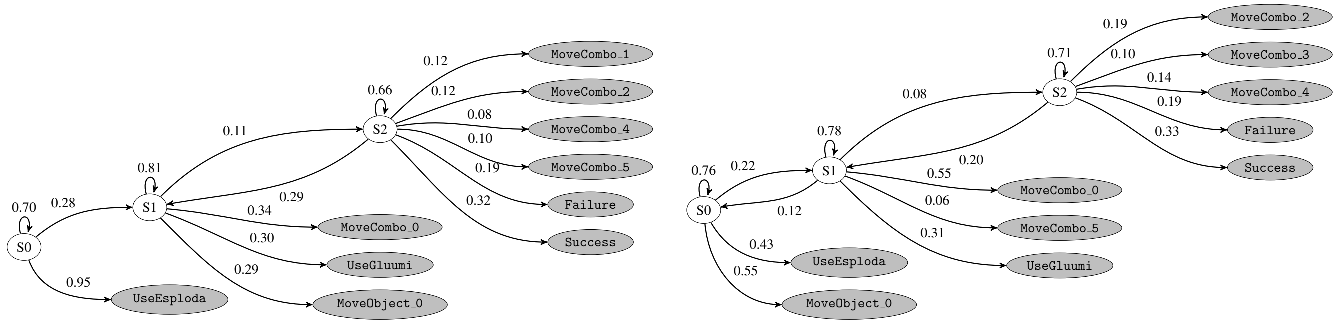
Figure 5 compares the true post-test scores against the predicted values. The regression line is calculated using LASSO and the 95% confidence interval is plotted. The regression model also provides us with an insight into how important each game level is in distinguishing between high- and low-performing students.

Table 3 shows the weights of each game level in the best performing 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 supports that the difference between the sequence of student interactions in this game level has the highest factor in distinguishing between high- and low-performing students.

The difference in predictive performance between game levels is likely related to game design. Some game levels are designed to introduce game tools and techniques while others are created to present choices to players aligned with math proficiency. We believe the source of difference is mainly due to the fact that finding the solution to later game levels requires mastery in all three major subskills targeted by the game design.

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.



(a) High performing students

(b) Low performing students

Figure 6: Transition diagram learned from data of students playing one of the games

4.4 Discussion of Interpretability

Predicting post-test scores is not the only motivation for designing the pipeline. We are also looking for an interpretable model that can give us insight into how students interact with the system and identify patterns that are suggestive of learning. 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- and low-performing students and removed the edges that had a 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 `MoveCombo_ClusterID` until 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 an object into the outlier cluster (`MoveObject_0 and MoveCombo_0`) is also higher across states for low-performing students.

5. RELATION TO PRIOR WORK

The potential of computer games for educational purposes has been of interest since nearly the beginning of video games. 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. However, games have produced mixed learning outcomes in the classroom [10, 15] and studying student interactions is a crucial step toward understanding their learning process and improving the game environment in the future.

There have been numerous attempts by the educational research community to develop analytic methods and build predictive models based on the data from educational games. For example, *Rumble Blocks* [3] is a physics 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. [9] learned a two-dimensional context-free grammar (CFG) from student interactions within the game and used a modified version of CYK algorithm to parse and extract conceptual features from student solutions. They used the features and performed cluster analysis of student solutions to measure how many follow the designers' envisioned path. While the CFG framework offers a promising method to learn features from game activity logs, we are particularly interested in a model that can represent the sequence of interactions and not the characteristics of final submission.

Crystal Island [20] is an educational 3D game environment for learning microbiology in middle school. Rowe and Lester [19] used a hand-authored dynamic Bayesian network (DBN) to model student knowledge in four broad components: narrative, content, strategy, and solution knowledge. They handpicked a set of observable actions that can serve as evidence of the student's underlying knowledge in each category and defined the parameters of the model in terms of conditional probability tables for each of the network nodes (observable and latent). They used observations from 116 eighth-graders interacting with the game environment in 50-minute sessions. In order to evaluate the model, they experimented with different threshold levels to discriminate between mastered and un-mastered knowledge components and used the final estimate of student knowledge in each category (mastered vs. un-mastered) to predict their post-test results. The DBN framework is an example of extensive domain authoring that we are trying to avoid by using data-driven methods.

One way to avoid extensive domain authoring is to use the state-action graph of the game environment. States are different configurations of the game environment and actions represent different ways students can interact with the game in each state. For example, *Refraction* [1] is another educational game for learning about fractions by splitting laser beams into fractional amounts to target spaceships and avoid asteroids. Liu et al. [14] 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 the last game level based on the history of movements in previous game levels. Lee et al. [13] extended the former framework by building a 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 [14]. While the state-action graph of the game environment offers a potential for micro-level analysis of student interactions, we are particularly interested in patterns that are representative of student proficiency.

Quantum Spectre is a more complex puzzle-style educational game designed for high school students to learn about optics. Eagle et al. [5] used interaction networks (IN) to visualize and derive insight into students' problem-solving behavior and common misconceptions. They used data from 195 students in 15 classes and created the full IN of every state and possible actions for each game level. Next, they used clustering to reduce the state-action space into a high-level visualization of how students progress through the puzzle and classify each student interaction into four categories: correct, placement error, rotation error, and puzzle error. Finally, they picked a smaller set of students (n=10) and used a linear regression model for predicting their post-test results. The results show a direct negative relationship between science-related gameplay errors and the post-test scores. Constructing the full interaction network for each game level can easily become intractable for complex games such as *Alice in AreaLand*. Moreover, integrating results from multiple game levels is not straightforward since each game level has its own state-action graph and further engineering is required to extract features that are generalizable across game levels.

6. CONCLUSIONS AND FUTURE WORK

Educational games are an interesting source of data that provide evidence of complex cognitive behavior. In this study, we propose a novel data-driven method, SPRING, for student modeling in educational games. Modeling student behavior is a promising research direction with applications for game designers, educators, and even policy makers.

Unlike prior work in invisible assessment, we do not use domain knowledge to design features that relate gameplay behavior with student performance. Instead, we explore the effectiveness of data-driven methods for student modeling in educational games. We evaluate our models quantitatively in terms of how well it can predict student assessments. To demonstrate the effectiveness of our framework, we predict the post-test scores of students using a regression model. The parameters of the regression model quantify how much an educational game is useful in distinguishing between high- and low-performing students. We suggest that SPRING is an interpretable model, and it may provide insight into how students learn.

There are many possible directions for future work. On a lower level, we can integrate the discretization and sequence modeling step by employing Hidden Markov Models that accept slot and filler sequences as input. On a higher level, instead of dividing students into two groups (high- and low-performing), we can use the HMM to cluster students [2, 25] into more groups that might be more representative of different approaches students follow to solve each game level. We can also replace our regression model with a more powerful ensemble method that considers different subskills in order to integrate features of each game level to predict the post-test scores. Future work could address the assumptions of our method. For example, SPRING assumes that events are always described with the same set of slots and each slot can only accept one filler.

Even though we only evaluated SPRING using data from a single game, the algorithm may be useful for similar gaming environments that use slot-and-filler structures to log student interactions with the system. We are optimistic that our approach may be useful to make assessment less intrusive and better embedded to classroom education.

7. REFERENCES

1. Erik Andersen, Yun-En Liu, Ethan Apter, François Boucher-Genesse, and Zoran Popović. 2010. Gameplay analysis through state projection. In *Proceedings of the fifth international conference on the foundations of digital games*. ACM, 1–8.
2. Manuele Bicego, Vittorio Murino, and Mário AT Figueiredo. 2003. Similarity-based clustering of sequences using hidden Markov models. In *Machine learning and data mining in pattern recognition*. Springer, 86–95.
3. Michael G Christel, Scott M Stevens, Bryan S Maher, Sean Brice, Matt Champer, Luke Jayapalan, Qiaosi Chen, Jing Jin, Daniel Hausmann, Nora Bastida, and others. 2012. RumbleBlocks: Teaching science concepts to young children through a Unity game. In *Computer Games (CGAMES), 2012 17th International Conference on*. IEEE, 162–166.
4. Bi-Ru Dai, I Lin, and others. 2012. Efficient map/reduce-based dbscan algorithm with optimized data partition. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 59–66.
5. Michael Eagle, Elizabeth Rowe, Drew Hicks, Rebecca Brown, Tiffany Barnes, Jodi Asbell-Clarke, and Teon Edwards. 2015. Measuring Implicit Science Learning with Networks of Player-Game Interactions. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. ACM, 499–504.
6. 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.
7. 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.
8. 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.

9. Erik Harpstead, Christopher J MacLellan, Kenneth R Koedinger, Vincent Aleven, Steven P Dow, and Brad Myers. 2013a. Investigating the solution space of an open-ended educational game using conceptual feature extraction. (2013).

10. Erik Harpstead, Brad A Myers, and Vincent Aleven. 2013b. In search of learning: facilitating data analysis in educational games. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 79–88.

11. Peter Hofman, Bryan Goodwin, and Stuart Kahl. 2015. Re-Balancing Assessment: Placing Formative and Performance Assessment at the Heart of Learning and Accountability. (2015).

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

13. 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*.

14. 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*.

15. Merrilea J Mayo. 2009. Video games: A route to large-scale STEM education? *Science* 323, 5910 (2009), 79–82.

16. 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.

17. Mostofa Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok Choudhary. 2012. A new scalable parallel DBSCAN algorithm using the disjoint-set data structure. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. IEEE, 1–11.

18. 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.

19. Jonathan P Rowe and James C Lester. 2010. Modeling User Knowledge with Dynamic Bayesian Networks in Interactive Narrative Environments.. In *AIIDE*.

20. Jonathan P Rowe, Lucy R Shores, Bradford W Mott, and James C Lester. 2010. Integrating learning and engagement in narrative-centered learning environments. In *Intelligent Tutoring Systems*. Springer, 166–177.

21. 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.

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

23. 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.

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

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

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