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

**Mohammad H. Falakmasir***\*,*# **Jose´ P. Gonza´ lez-Brenes***\** **Geoffrey J. Gordon**§ **Kristen E. DiCerbo***\**

*\** School Research

Pearson

*{*jose.gonzalez-brenes, kristen.dicerbo*}*@pearson.com

# Intelligent Systems Program University of Pittsburgh [f](mailto:falakmasir@pitt.edu)[alakmasir@pitt.edu](mailto:alakmasir@pitt.edu)

§ Machine Learning Carnegie Mellon University [ggordon@cs.cmu.edu](mailto:ggordon@cs.cmu.edu)

**ABSTRACT**

Student assessments are important because they allow collect- ing evidence about learning. However, time spent on evaluat- ing students may be otherwise used for instructional activities. Computer-based learning platforms provide the opportunity for unobtrusively gathering students’ digital learning foot- prints. 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 do- main knowledge engineering. Moreover, it produces a simple interpretable model that not only fits the data, but also predics learning outcomes. We validate our framework using data col- lected 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 in- stead 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 be- tween pre-K and highschool [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].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org.](mailto:permissions@acm.org)

*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>

Learning and succeeding is not easily measured by solving constrained tasks and answering verbal or multiple-choice questions. Advancements in educational measurements al- low us to diagnose student competencies more accurately and effectively at various levels during the course of learning. Ac- cording 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 identi- fying, 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 educa- tional games is a promising way to perform invisible assess- ment. 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 assess- ment 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 do- main 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 *Evi- dence 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 makes 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.

**2. BACKGROUND**

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. Typically each action or event has a set of slots (properties). Each slot may accept zero to multiple fillers. Each filler represents a property value of the slot. 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.

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.

**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 *{}*

**Procedure 1** The SPRING algorithm

**Input:** : A log file **Lg***,***s***,***t** of slot and filler structure for each game *g* and student *s* (indexed by time *t* ), results from a student assessment *ys* , number of performance clusters *K*, number of latent states *H* .

1: **for** each game *g* **do**

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

2: **Dg***,***s***,***t** *←* Discretize Fillers(**Lg***,***s***,***t** )

3: *(s, c) ←* Cluster Students(*ys , K*)

4: **for** each student performance group *c* **do**

5: **Dc** *←* **Dg***,***s***,***t** where *s ∈ (s, c)*

6: *θg,c ←* Learn HMM(**Dc** *, H* )

7: **for** each sequence *d* in **Dg***,***s do**

8: *φg,s ←* Extract Features(*d, θg,c* )

9: Learn a model that predicts *ys* from *φg,s*

Discretization

Sequence Model

Feature

Vector Model

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 (in slot-and-filler format) into feature vec- tors. Feature vectors need to be of a predetermined dimension, and a game log may have a variable number of slots.

Machine learning algorithms that are able to work with data that is not represented as feature vectors include kernel meth- ods and graphical models. Popular general-purpose graphical models, such as Bayesian Networks and Hidden Markov Mod- els (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 dimen- sions for each time step. This is not clear how to transform observations when the number of fillers can vary for different slots. For example, in Table 1, the Move Object requires fillers with 2 dimensions, while the Game Start has no fillers. On the other had, *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

Procedure 1 describes the three main steps of our pipeline: dis-

cretization, sequence modeling, and feature vector modeling. The discretization step transforms slot and filler observations into multinomial indicators that are used as evidence for learn- ing. 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 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*

Procedure 2 describes the method we use to discretize fillers for each game. The input is time-ordered student interactions in slot-and-filler format for a single game *g*. The purpose is to transform the slot and filler events into discrete (multinomial) observations appropriate for sequence modeling. For example, consider the Move Object event from Table 1. We wish to transform the different possible fillers into discrete units. Fig- ure 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.

In order to transform the slot and filler events into discrete observations, we use an unsupervised clustering algorithm for

**Procedure 2** The Discretization Step of SPRING



**Input:** : *Lg,s,t* , sequence of slot-and-filler observations in game level *g*

1: **procedure** DISCRETIZE FILLERS(**Lg***,***s***,***t** )

2: **Z** *←* enumerate slots(**Lg***,***s***,***t** )

3: **for** each slot *z* **do**

4: **Fz** *←* get fillers(**Lg***,***s***,***t** )

5: *(***Fz** , clustersIDs *) ←* Cluster(**Fz** )

6: model *←* learn classifier(**Fz** , clustersIDs)

7:

8: **for** each student *sf* **do**

9: **for** each time step *t f* **do**

10: **if** *z* == slot(*Lg,sf,t f* ) **then**

11:

*f*ˆ *←* model.predict(filler(*Lg,sf,t f* ))

12: *d ←* concat(z, *f*ˆ)

13: **Dg***,***s***f,***t***f ← d*

14: **return D***g,s,t*

the fillers of each slot. For example, in Figure 1b, we have three clusters, cluster one and two (red and yellow respec- tively) 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 slot and cluster labels as a observations in the sequence modeling phase. For example, instead of modeling from an infinite domain like

“Move Object *(x* : 363*, y* : 83*)*”, we use a number of discrete

observations like “Move Object Cluster 1”.

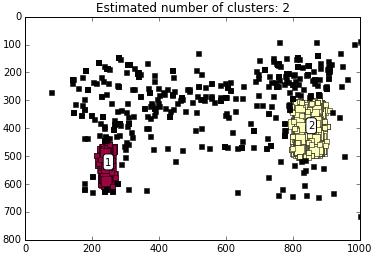
The specific clustering algorithm we used in our experiments is DBSCAN [6]. In preliminary experiments we tried other unsupervised methods, but we found DBSCAN easier to use because it can find arbitrarily shaped clusters and does not require one to specify the number of clusters in advance. The time 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].

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 par- ticular, 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 clus- ter 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*

In order to find an interpretable model that describes the be- havior of different groups of students, researchers often use unsupervised and non-parametric methods. However, since we are particularly trying to find patterns among students that are predictive of their post-test performance, we use the median

(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.

of post-test scores to divide students into two groups. In each game level, we place the students who received a post-test score below or equal to the median in the low-performing group and the students who received a post-test score higher than the median in the high-performing group. Due to the small number of students in our dataset, particularly in the later game levels, we only use two clusters. We hypothesize that students in different 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 statis- tical 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 sta- tistical patterns that can describe and distinguish those states.

Learning the “best” value for 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 pro- cess* 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 highperforming 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**

Given a sequence of student data, we can determine how likely it is for the student to be in the high-performing or low- performing group. We calculate this likelihood by estimating the Forward-Backward probabilities on each student sequence of actions based on the two HMMs models. Then, for each student *s* in each game level *g*, we calculate the difference (*ds,g* ) between the log-likelihood of observing the sequence of actions given the two models. Since we are learning different HMMs for each game level, we use a hyperbolic tangent function in order to convert the difference to a value between

-1 and 1:

*ds,g* = *t anh*[*K ∗* (*θg,*high *− θg,*low )] (1) This step simply scales and shifts the difference between log-

likelihoods in a way that if one HMM fits the data much better than the other, the value of *ds,g* is closer to one of the bound- aries and when there is no clear difference the value would be close to zero. The parameter K represents the scaling factor. Larger values for K result in output distribution that are closer to +1 and -1 in case of clear difference. We experimented with different K values across game levels and picked K=100.

In the next step, we use the transformed log-likelihood dif- ferences in each game level as features of a linear regression model that predicts the post-test scores:

*y*ˆ*s* (*β* ) = ∑ *βg · ds,g* + *β*0 (2)

*g*

Here, *β*0 is an intercept for the model. We optimized the pa- rameters of the model using a 5-fold cross validation on our development set. We experimented with different regulariza- tion methods, but only report LASSO [26] as it worked best in our preliminary comparisons:

*β ∗* = argmin *||ys − y*ˆ*s* (*β* )*||*2 + *λ · ||β ||*1 (3)

*β*



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.

**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 System1 . It focuses on teaching and assessing geometric mea- surement, specifically the understanding of area, among 6th grade students. The game targets three main stages in the de- velopment 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 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 re- quest 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 itera- tion 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,

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

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

16000

250

14000

12000

200

10000

8000

6000

4000

2000

Use Multi Use Esploda Use Gluumi Hint

Go Alice

Combo Move

Object Moved

150

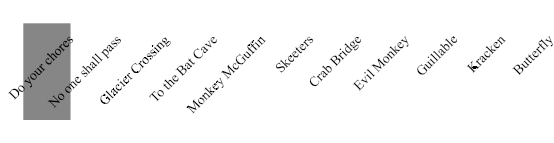
**Sequence Length**

100

50

0 0

Figure 3: Frequency of Events in Each Game Level



they either advance to the next level or receive feedback and stay in 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 stu- dents 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 de- scribed above. The post-test scores ranges 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 first attempt and change their answers multiple times to find the solution. Different attempts are a rich data source for studying students’ change of perfor- mance 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

Figure 4: Boxplot of Sequence Length in Each Game Level

sequences of multinomial variables that can be used as evi- dence 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 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 descretized observa- tions 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 Hierar- chical Dirichlet Process [7].

Additionally, we evaluate two baseline methods:

*•* Success / Failure: A regression model with eleven binary features. Each feature represents whether a student success- fully passed each of the 11 different mini-games on the first attempt.

*•* Sequence Length: A regression model that uses the normal- ized 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 evalu- ation 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 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 suggests that even though there is a positive cor- relation between the true and SPRING’s predicted post-test scores, the improvement in terms of MAE is not conclusive. This effect is clearly observable in Table 2 where a 0.4 im- provement in correlation over the first baseline resulted in 0.09 and 0.26 improvement in MAE and RMSE respectively. It is unclear which evaluation metric is better for our application. 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.

Figure 5 compares the true post-test scores against the pre- dicted 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 per- forming 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 differ- ence between the sequence of student interactions in this game

**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.

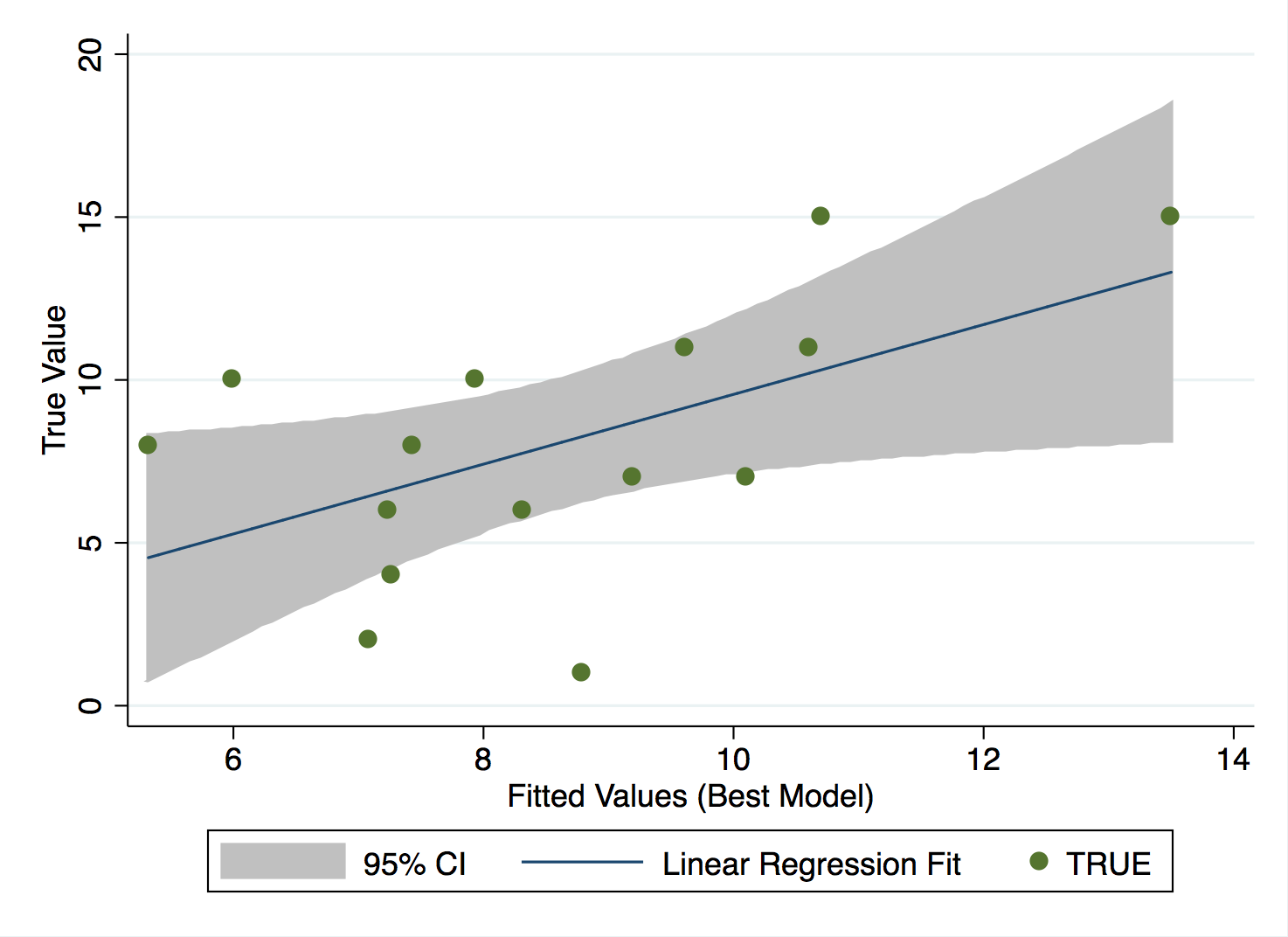


Figure 5: Results of predicting the post-test scores of the held-out set along with the regression line and 95% interval.

level has the highest factor in distinguishing between high- and low-performing students.

**4.4 Discussion and Limitations**

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 fea- tures from it to predict the post-test results using a regression model and suggested improvement over two baselines. As reported in the results section, the SPRING predictions while significantly correlated with true post-test scores, did not lead to significant improvement in MAE. We investigated this prob- lem in more depth and we believe that the reason we could not achieve a significant improvement was mainly because we evaluated the algorithm on a small dataset. Our held-out set contains only 15 students and a few of them achieved poor performance results in the post-test while successfully finish- ing most of the game levels. This observation could raise a question for the utility of similar data-driven models. However, we believe that using a larger data set will lower the negative effect of such students in the future.

Another limitation of our approach is that our student perfor- mance clustering phase uses the post-test scores (independent variable) to divide students into high- and low-performance groups. In order to avoid using information from the indepen- dent 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.

The difference in predictive performance between game lev- els 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.

0.81

0.11

0.66

S2

0.12

0.12

0.08

0.10

0.19

MoveCombo 1

MoveCombo 2

MoveCombo 4

MoveCombo 5

0.76

0.22

0.78

S1

0.08

0.20

0.55

0.71

S2

MoveCombo 0

0.19

0.10

0.14

0.19

0.33

MoveCombo 2

MoveCombo 3

MoveCombo 4

Failure

Success

0.70

S0

0.28

0.95

S1

UseEsploda

0.29

0.34

0.30

0.29

MoveCombo 0

UseGluumi

MoveObject 0

0.32

Failure

Success

S0

0.43

0.55

0.12

UseEsploda

MoveObject 0

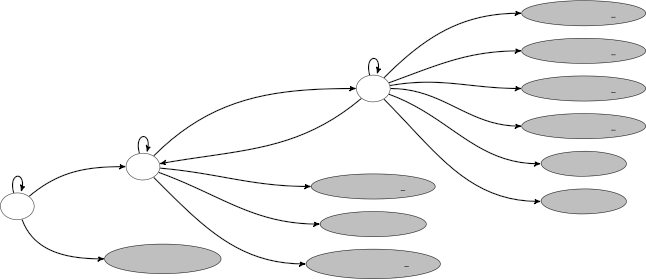
0.06

0.31

MoveCombo 5

UseGluumi

(a) High Performing



(b) Low Performing

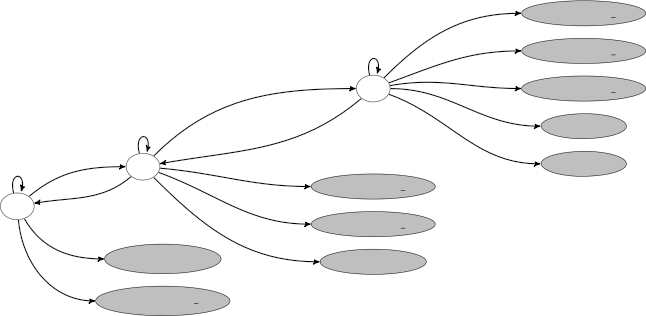


Figure 6: Learned Transition Models for One of the Game Levels

On the other hand, predicting post-test scores was not the only reason for designing the pipeline. We also seek for a descriptive model that can give us insight into low-level pat- terns 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 students 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- and low-performing students and removed the edges that had 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 illus-

trated in Figure 6a, high-performing students follow the ex- pected 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 success- fully 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 (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 inter- actions is a crucial step toward understanding their learning process and improving the game environment 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. For example, *Rumble Blocks* [3] is a physics educational game designed to teach basic concepts of structural stability and bal- ance 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 con- ceptual 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 offer a promising method to learn features from game activity logs, we are particularly interested in a model that can represent 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 hand picked 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 thresh- old 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 pre- dict 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 dif- ferent 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 col- laborative 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 to predict their post-test results. The results show a direct negative relationship between science-related game play 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 straight forward 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 pro- vide 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 in open-ended environments such as educational games is a promising direction with applications for game designers, educators and even policy makers.

We evaluate our models quantitatively in terms of how well we can predict student assessments. We suggest that the mod- els learned can be interpreted and provide insight into how students learn. Additionally, the parameters of the regression model quantify how much an educational game is useful 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 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.

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 ap- proach may be useful to make assessment less intrusive and better embedded to classroom education.

**7. REFERENCES**

1. Erik Andersen, Yun-En Liu, Ethan Apter, Franc¸ois Boucher-Genesse, and Zoran Popovic´. 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 Ma´rio 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, Jo¨ rg 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 Lazar´ın. 2014. Testing overload in America’s schools.

*Center for American Progress. Retrieved from* [*www.americanprogress.org/issues/education/report/*](http://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.