Detecting Student Misuse of Intelligent Tutoring Systems

Ryan Shaun Baker, Albert T. Corbett, Kenneth R. Koedinger

Human-Computer Interaction Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA, 15217, USA
{rsbaker, corbett, koedinger}@cmu.edu

Abstract. Recent research has indicated that misuse of intelligent tutoring software is correlated with substantially lower learning. Students who frequently engage in behavior termed "gaming the system" (behavior aimed at obtaining correct answers and advancing within the tutoring curriculum by systematically taking advantage of regularities in the software's feedback and help) learn only 2/3 as much as similar students who do not engage in such behaviors. We present a machine-learned Latent Response Model that can identify if a student is gaming the system in a way that leads to poor learning. We believe this model will be useful both for re-designing tutors to respond appropriately to gaming, and for understanding the phenomenon of gaming better.

1 Introduction

There has been growing interest in the motivation of students using intelligent tutoring systems (ITSs), and in how a student's motivation affects the way he or she interacts with the software. Tutoring systems have become highly effective at assessing what skills a student possesses and tailoring the choice of exercises to a student's skills [6,14], leading to curricula which are impressively effective in real-world classroom settings [7]. However, intelligent tutors are not immune to the motivational problems that plague traditional classrooms. Although it has been observed that students in intelligent tutoring classes are more motivated than students in traditional classes [17], students misuse intelligent tutoring software in a way that suggests less than ideal motivation [1,15]. In one recent study, students who frequently misused tutor software learned only 2/3 as much students who used the tutor properly, controlling for prior knowledge and general academic ability [5]. Hence, intelligent tutors which can respond to differences in student motivation as well as differences in student cognition (as proposed in [9]) may be even more effective than current systems.

Developing intelligent tutors that can adapt appropriately to unmotivated students depends upon the creation of effective tools for assessing a student's motivation. Two different visions of motivation's role in intelligent tutors have resulted in two distinct approaches to assessing motivation. In the first approach, increased student motivation is seen as an end in itself, and the goal is to create more empathetic, enjoyable,

and motivating intelligent tutoring systems. In order to do this, it is desirable to have the richest possible picture of a student's current motivational state – for instance, de Vicente and Pain have developed a model that classifies a student's motivational state along 9 axes [8].

An alternate approach focuses on motivation as a factor which affects learning; improving motivation is viewed primarily as a means to improving learning. Investigating motivation in this fashion hinges upon determining which motivation-related behaviors most strongly affect learning, and then understanding and assessing those specific behaviors and motivations. For instance, Mostow and his colleagues have identified that some students take advantage of learner-control features of a reading tutor to spend the majority of their time playing rather than working, or to repeatedly re-read stories they already know by heart [15]. Another motivation-related behavior is "help abuse", where a student quickly and repeatedly asks for help until the tutor gives the student the correct answer, often before the student attempts the problem on his or her own [18]. Aleven and Koedinger have determined that seeking help before attempting a problem on one's own is negatively correlated to learning, and have worked to develop a model of student help-seeking that can be used to give feedback to students on how to use help more effectively [1].

In [5], we presented a study on a category of strategic behavior, termed "gaming the system", which includes some of the motivation-related behaviors discussed above. Gaming the system is behavior aimed at performing well in an educational task by systematically taking advantage of properties and regularities in the system used to complete that task, rather than by thinking about the material. Students in our study engaged in two types of gaming the system: help abuse and systematic trial-and-error. We investigated these phenomena by observing students for two class periods as the students used a tutor lesson on scatterplot generation, using methods adapted from past quantitative observational studies of student off-task behavior in traditional classrooms [cf. 12]. Each student's behavior was observed a number of times during the course of each class period. The students were observed in a specific order determined before the class began in order to prevent bias towards more interesting or dramatic events. In each observation, each student's behavior was coded as being one of the following categories: working in the tutor, talking on-task, talking off-task, silently off-task (for instance, surfing the web), inactive (for instance, asleep), and gaming the system.

We found that a student's frequency of gaming was strongly negatively correlated with learning, but was *not* correlated with the frequency of other off-task behavior; nor was other off-task behavior significantly correlated with learning, suggesting that not all types of low motivation are equivalent in their effects on student learning with ITSs. The evidence from this study was neutral as to whether gaming was harmful in and of itself (by hampering the learning of the specific skills gamed) or whether it was merely symptomatic of non-learning goals [cf. 3].

Understanding why students game the system will be essential to deciding how the system should respond. Ultimately, though, whatever remediation approach is chosen, it is likely to have costs as well as benefits. For instance, preventive approaches, such as changing interface widgets to make them more difficult to game or delaying suc-

cessive levels of help to prevent rapid-fire usage[1], may reduce gaming, but at the cost of making the tutor more frustrating and less time-efficient for other students. Since many students use help effectively [18] and seldom or never game the system [5], the costs of using such an approach indiscriminately may be higher than the rewards. Whichever approach we take to remediating gaming the system, the success of that approach is likely to depend on accurately and automatically detecting which students are gaming the system and which are not.

In this paper, we report progress towards this goal: we present and discuss a machine-learned Latent Response Model (LRM) [13] that is highly successful at discerning which students frequently game the system in a way that is correlated with low learning. Cross-validation shows that this model should be effective for other students using the same tutor lesson. Additionally, this model corroborates the hypothesis in Baker et al 2004 that students who game the system (especially those who show the poorest learning) are more likely to do so on the most difficult steps.

2 Methods

2.1 Data Sources

In order to develop an algorithm to detect that a student is gaming the system, we combined three sources of data on student performance and behavior in a cognitive tutor lesson teaching about scatterplot generation [4]. All data was drawn from a group of 70 students using that cognitive tutor lesson as part of their normal mathematics curricula.

The first source of data was a log of every action each student performed while using the tutor. Each student performed between 71 and 478 actions within the tutor. For each action, we distilled 24 features from the log files. The features were:

- The tutoring software's assessment of the action – was the action correct, incorrect and indicating a known bug (procedural misconception), incorrect but not indicating a known bug, or a help request[2]? (represented as 3 binary variables)
- The type of interface widget involved in the action – was the student choosing from a pull-down menu, typing in a string, typing in a number, plotting a point, or selecting a checkbox? (represented as 4 binary variables)
- The tutor's assessment, post-action, of the probability that the student knew the skill involved in this action, called "pknow" (derived using the Bayesian knowledge tracing algorithm in [6]).
- Was this the student's first attempt to answer (or get help) on this problem step?

[1] A modification currently in place in the commercial version of Cognitive Tutor Algebra.

[2] Due to an error in tutor log collection, we only obtained data about entire help requests, not about the internal steps of a help request.

- "Pknow-direct", a feature drawn directly from the tutor log files (the previous two features were distilled from it). If the current action is the student's first attempt on this problem step, then pknow-direct is equal to pknow, but if the student has already made an attempt on this problem step, then pknow-direct is -1. Pknow-direct allows a contrast between a student's first attempt on a skill he/she knows very well and a student's later attempts.
- How many seconds the action took (both the actual number of seconds, and the standard deviations from the mean time taken by all students on this problem step across problems.)
- How many seconds were spent in the last 3 actions, or 5 actions. (two variables)
- How many seconds the student spent on each opportunity to practice this skill, averaged across problems.
- The total number of times the student has gotten this specific problem step wrong, across all problems. (includes multiple attempts within one problem)
- The number of times the student asked for help or made errors at this skill, including previous problems.
- How many of the last 5 actions involved this problem step.
- How many times the student asked for help in the last 8 actions.
- How many errors the student made in the last 5 actions.

The second source of data was the set of human-coded observations of student behavior during the lesson. This gave us the approximate proportion of time each student spent gaming the system, $G_0 \ldots G_{69}$.

Since it is not clear that all students game the system for the same reasons or in exactly the same fashion, we used student learning outcomes as a third source of data. We divided students into three sets: a set of 53 students never observed gaming the system, a set of 9 students observed gaming the system who were not obviously hurt by their gaming behavior, having either a high pretest score or a high pretest-posttest gain (this group will be referred to as GAMED-NOT-HURT), and a set of 8 students observed gaming the system who were apparently hurt by gaming, scoring low on the post-test (referred to as GAMED-HURT). It is important to distinguish GAMED-HURT students from GAMED-NOT-HURT students, since these two groups may behave differently (even if an observer sees their actions as similar), and it is more important to target interventions to the GAMED-HURT group than the GAMED-NOT-HURT group. This sort of distinction has been found effective for developing algorithms to differentiate cheating from other categories of behavior [11].

2.2 Data Modeling

Using these three data sources, we trained a density estimator to predict how frequently an arbitrary student gamed the system. The algorithm we chose was forward-selection [16] on a set of Latent Response Models (LRM) [13]. LRMs provide two prominent advantages for modeling our data: First, they offer excellent support for integrating multiple sources of data, including both labeled and unlabeled data. Sec-

ondly, an LRM's results can be interpreted much more easily by humans than the results of most neural network, support vector machine, or decision tree algorithms, facilitating thought about design implications.

The set of possible parameters was drawn from linear effects on the 24 features discussed above (parameter*feature), quadratic effects on those 24 features (parameter*feature^2), and 23x24 interaction effects between features (parameter* feature$_A$*feature$_B$). During model selection, the potential parameter was added that most reduced the mean absolute deviation between our model predictions and the original data, using iterative gradient descent to find the best value for each candidate parameter. Forward-selection continued until no parameter could be found which appreciably reduced the mean absolute deviation. The best-fitting model had 4 parameters, and no model considered had more than 6 parameters.

Given a specific model, the algorithm first predicted whether each individual tutor action was an instance of gaming the system or not. Given a set of n parameters across all students and actions, with each parameter α_i associated with feature X_i (or X_i^2, or X_i Y_i), a prediction P_m as to whether action m was an instance of gaming the system was computed as $P_m = \alpha_0 X_0 + \alpha_1 X_1 + \alpha_2 X_2 + \ldots + \alpha_n X_n$. Each prediction P_m was then thresholded using a step function, such that if $P_m \leq 0.5$, $P'_m = 0$, otherwise $P'_m = 1$. This gave us a classification P'_m for each action within the tutor. We then determined, for each student, what proportion of that student's actions P'_m were classified as gaming, giving us a set of values $G'_0 \ldots G'_{69}$. By comparing the values $G'_0 \ldots G'_{69}$ to the observed proportions of time each student spent gaming the system, $G_0 \ldots G_{69}$, we computed each candidate model's deviation from the original data. These deviations were used during iterative gradient descent and model selection, in order to find the best model parameters.

Along with finding the best model for the entire data set, we conducted Leave One Out Cross Validation (LOOCV) to get a measure of how effectively the model will generalize to students who were not in the original data set (the issue of how well the model will generalize to different tutor lessons will be discussed in the Future Work section). In doing a LOOCV, we fit to sets of 69 of the 70 students, and then investigated how good the model was at making predictions about the 70th student.

2.3 Classifier

For the purpose of assigning interventions, we developed a classifier to identify which students are gaming and in need of an intervention. We did so by setting a threshold on how often the model perceives a student is gaming. Any student above this threshold is considered to be gaming, and all other students are considered not gaming. Given different possible thresholds, there is a tradeoff between correctly identifying gaming students (hits) and incorrectly identifying non-gaming students as gaming students (false positives), shown in the Receiver Operating Characteristic (ROC) curve in Figure 1. The classifier's ability to distinguish gaming is assessed with an A' value, which gives the probability that if the model is given one gaming student and one non-gaming student, it will accurately identify which is which [10].

3 Results

3.1 Our Classifier's Ability to Detect Gaming Students

In this section, we discuss our classifier's ability to detect which students game. All discussion is with reference to the cross-validated version of our model/classifier, in order to assess how well our approach will generalize to the population in general, rather than to just our sample of 70 students.

Since most potential interventions will have side-effects and costs (in terms of time, if nothing else), it is important both that the classifier is good at correctly identifying the GAMED-HURT students who are gaming and not learning, and that it rarely assigns an intervention to students who do not game.

If we take a model trained to treat both GAMED-HURT and GAMED-NOT-HURT students as gaming, it is significantly better than chance at classifying the GAMED-HURT students as gaming ($A' = 0.82$, $p < 0.001$). At the threshold value with the highest ratio between hits and false positives, this classifier correctly identifies 88% of the GAMED-HURT students as gaming, while only classifying 15% of the non-gaming students as gaming. Hence, this model can be reliably used to assign interventions to the GAMED-HURT students. By contrast, the same model is not significantly better than chance at classifying the GAMED-NOT-HURT students as gaming ($A' = 0.57$, $p = 0.58$).

Since it is more important to detect GAMED-HURT students than GAMED-NOT-HURT students, it is conceivable that there may be extra leverage gained from

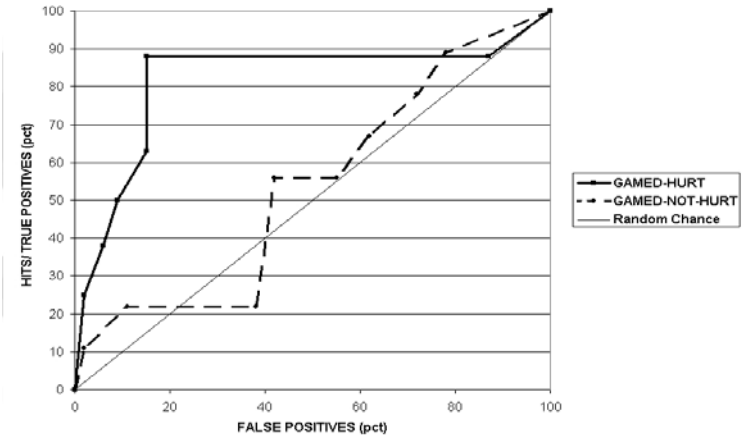

Fig. 1. Empirical ROC Curves showing the trade-off between true positives and false positives, for the cross-validated model trained on both groups of gaming students.

training a model only on GAMED-HURT students. In practice, however, a model trained only on GAMED-HURT students (A' =0.77) does no better at identifying the GAMED-HURT students than the model trained on both groups of students. Thus, in our further research, we will use the model trained on both groups of students to identify GAMED-HURT students.

It is important to note that although gaming is negatively correlated to post-test score, our classifier is not just classifying which students fail to learn. Our model is not better than chance at classifying students with low post-test scores (A' =0.60, p=0.35) or students with low learning (low pre-test *and* low post-test) (A' =0.56, p=0.59). Thus, our model is not simply identifying all gaming students, nor is it identifying all students with low learning – it is identifying the students who game *and* have low learning: the GAMED-HURT students.

3.2 Describing Our Model

At this point, our primary goal for creating a model of student gaming has been achieved – we have developed a model that can accurately identify which students are gaming the system, in order to assign interventions. Our model does so by first predicting whether each of a student's actions is an instance of gaming. Although the data from our original study does not allow us to directly validate that a specific step is an instance of gaming, we can investigate what our model's predictions imply about gaming, and whether those predictions help us understand gaming better.

The model predicts that a specific action is an instance of gaming when the expression shown in Table 1 is greater than 0.5.

Feature F_0,"ERROR-NOW, MANY-ERRORS-EACH-PROBLEM", identifies a student as more likely to be gaming if the student has already made at least one error on this problem step within this problem, and has also made a large number of errors on this problem step in previous problems. It identifies a student as less likely to be gaming if the student has made a lot of errors on this problem step in the past, but now probably understands it (and has not yet gotten the step wrong in this problem).

Table 1. How the model predicts whether a specific action is an instance of gaming

Name	Coefficient	Feature
F_0: "ERROR-NOW, MANY-ERRORS-EACH-PROBLEM"	-0.0375	pknow-direct * number of errors the student has made on this problem step (across all problems)
F_1: "QUICK-ACTIONS-AFTER-ERROR"	+ 0.094	pknow-direct * time taken, in SD above (+) or below (-) the mean time for all students, on this problem step (across all problems)
F_2: "MANY--ERRORS-EACH-PROBLEM-POPUP"	+ 0.231	number wrong on this problem step (across all problems), if the problem step uses a popup menu
F_3: "SLIPS-ARE-NOT-GAMING"	- 0.225	pknow * how many errors the student made on last 5 actions

Feature F_1,"QUICK-ACTIONS-AFTER-ERROR", identifies a student as more likely to be gaming if he or she has already made at least one error on this problem step within this problem, and is now making extremely quick actions. It identifies a student as less likely to be gaming if he or she has made at least one error on this problem step within this problem, but works slowly during subsequent actions, or if a student answers quickly on his or her first opportunity (in a given problem step) to use a well-known skill.

Feature F_2,"MANY-ERRORS-EACH-PROBLEM-POPUP", indicates that making many errors across multiple problems is even more indicative of gaming if the problem-step involves a popup menu. In the tutor studied, popup menus are used for multiple choice questions where the responses are individually lengthy; but this enables a student to attempt each answer in quick succession.

Feature F_3,"SLIPS-ARE-NOT-GAMING", identifies that if a student has a high probability of knowing a skill, the student is less likely to be gaming, even if he or she has made many errors recently. Feature F_3 counteracts the fact that features F_0 and F_1 do not distinguish well-known skills from poorly-known skills, if the student has already made an error on the current problem step within the current problem.

The model discussed above is trained on all students, but is highly similar to the 70 models generated during cross-validation. Features F_0, F_2 and F_3 appear in over 97% of the cross-validated models, and feature F_1 appears in 71% of those models. No other feature was used in over 10% of the cross-validated models.

One surprising aspect of this model is that none of the features involve student use of help. We believe that this is primarily an artifact of the tutor log files we obtained; current research in identifying help abuse relies upon considerable data about the timing of each internal step of a help request (cf. [2]). Despite this limitation, it is interesting that a model can accurately detect gaming without directly detecting help abuse. One possibility is that students who game the system in the ways predicted by our model also game the system in the other fashions observed in our original study.

3.3 Further Investigations With Our Model

One interesting aspect of our model is how it predicts gaming actions are distributed across a student's actions. 49% of our model's 21,520 gaming predictions occurred in clusters where at least 2 of the nearest 4 actions were also instances of gaming. To determine the chance frequency of such clusters, we ran a Monte Carlo simulation where each student's instances of predicted gaming were randomly distributed across that student's 71 to 478 actions. In this simulation, only 5% (SD=1%) of gaming predictions occurred such clusters. Hence, our model predicts that substantially more gaming actions occur in clusters than one could expect from chance.

Our model also suggests that there is at least one substantial difference between when GAMED-HURT and GAMED-NOT-HURT students choose to game – and this difference may explain why the GAMED-HURT students learn less. Compare the model's predicted frequency of gaming on "difficult skills", which the tutor estimated the student had under a 20% chance of knowing (20% was the tutor's estimated prob-

ability that a student knew a skill upon starting the lesson), to the frequency of gaming on "easy skills", which the tutor estimated the student had over a 90% chance of knowing. The model predicted that students in the GAMED-HURT group gamed significantly more on difficult skills (12%) than easy skills (2%), $t(7)=2.99$, $p<0.05$ for a two-tailed paired t-test. By comparison, the model predicted that students in the GAMED-NOT-HURT group did not game a significantly different amount of the time on difficult skills (2%) than on easy skills (4%), $t(8)=1.69$, $p=0.13$. This pattern of results suggests that the difference between GAMED-HURT and GAMED-NOT-HURT students may be that GAMED-HURT students chose to game exactly when it will hurt them most.

4 Future Work and Conclusions

At this point, we have a model which is successful at recognizing students who game the system and show poor learning. As it has good results under cross-validation, it is likely that it will generalize well to other students using the same tutor.

We have three goals for our future work. The first goal is to study this phenomena in other middle school mathematics tutors, and to generalize our classifier to those tutors. In order to do so, we will collect observations of gaming in other tutors, and attempt to adapt our current classifier to recognize gaming in those tutors. Comparing our model's predictions about student gaming to the recent predictions about help abuse in [2] is likely to provide additional insight and opportunities. The second goal is to determine more conclusively whether our model is actually able to identify exactly when a student is gaming. Collecting labeled data, where we can link the precise time of each observation to the actions in a log file, will assist us in this goal. The third goal is to use this model to select which students receive interventions to reduce gaming. We have avoided discussing how to remediate gaming in this paper, in part because we have not completed our investigations into *why* students game. Designing appropriate responses to gaming will require understanding why students game.

Our long-term goal is to develop intelligent tutors that can adapt not only to a student's knowledge and cognitive characteristics, but also to a student's behavioral characteristics. By doing so, we may be able to make tutors more effective learning environments for all students.

Acknowledgements

We would like to thank Tom Mitchell, Rachel Roberts, Vincent Aleven, Lisa Anthony, Joseph Beck, Elspeth Golden, Cecily Heiner, Amy Hurst, Brian Junker, Jack Mostow, Ido Roll, Peter Scupelli, and Amy Soller for helpful suggestions and assistance. This work was funded by an NDSEG Fellowship.

References

1. Aleven, V., Koedinger, K.R. Investigations into Help Seeking and Learning with a Cognitive Tutor. In R. Luckin (Ed.), Papers of the AIED-2001 Workshop on Help Provision and Help Seeking in Interactive Learning Environments (2001) 47-58
2. Aleven, V., McLaren, B., Roll, I., Koedinger, K. Toward Tutoring Help Seeking: Applying Cognitive Modeling to Meta-Cognitive Skills. To appear at Intelligent Tutoring Systems Conference (2004)
3. Arbreton, A. Student Goal Orientation and Help-Seeking Strategy Use. In S.A. Karabenick (Ed.), Strategic Help Seeking: Implications For Learning And Teaching. Mahwah, NJ: Lawrence Erlbaum Associates (1998) 95-116
4. Baker, R.S., Corbett, A.T., Koedinger, K.R. Learning to Distinguish Between Representations of Data: a Cognitive Tutor That Uses Contrasting Cases. To appear at International Conference of the Learning Sciences (2004)
5. Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game the System". Proceedings of ACM CHI 2004: Computer-Human Interaction (2004) 383-390
6. Corbett, A.T. & Anderson, J.R. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User Modeling and User-Adapted Interaction Vol. 4 (1995) 253-278
7. Corbett, A.T., Koedinger, K.R., & Hadley, W. S. Cognitive Tutors: From the Research Classroom to All Classrooms. In P. Goodman (Ed.), Technology Enhanced Learning: Opportunities For Change. Mahwah, NJ : Lawrence Erlbaum Associates (2001) 235-263
8. de Vicente, A., Pain, H. Informing the Detection of the Students' Motivational State: an Empirical Study. In S. A. Cerri, G. Gouarderes, F. Paraguacu (Eds.), Proceedings of the Sixth International Conference on Intelligent Tutoring Systems (2002) 933-943
9. del Soldato, T., du Boulay, B. Implementation of Motivational Tactics in Tutoring Systems. Journal of Artificial Intelligence in Education Vol. 6(4) (1995) 337-376
10. Donaldson, W. Accuracy of d' and A' as Estimates of Sensitivity. Bulletin of the Psychonomic Society Vol. 31(4) (1993) 271-274.
11. Jacob, B.A., Levitt, S.D. Catching Cheating Teachers: The Results of an Unusual Experiment in Implementing Theory. To appear in Brookings-Wharton Papers on Urban Affairs.
12. Lloyd, J.W., Loper, A.B. Measurement and Evaluation of Task-Related Learning Behavior: Attention to Task and Metacognition. School Psychology Review vol.15(3)(1986) 336-345.
13. Maris, E. Psychometric Latent Response Models. Psychometrika vol.60(4) (1995) 523-547.
14. Martin, J., vanLehn, K. Student Assessment Using Bayesian Nets. *International* Journal of Human-Computer Studies vol. 42 (1995) 575-591
15. Mostow, J., Aist, G., Beck, J., Chalasani, R., Cuneo, A., Jia, P., Kadaru, K. A La Recherche du Temps Perdu, or As Time Goes By: Where Does the Time Go in a Reading Tutor that Listens? Sixth International Conference on Intelligent Tutoring Systems (2002) 320-329
16. Ramsey, F.L., Schafer, D.W. The Statistical Sleuth: A Course in Methods of Data Analysis. Belmont, CA: Duxbury Press (1997) Section 12.3
17. Schofield, J.W. Computers and Classroom Culture. Cambridge, UK: Cambridge University Press (1995)
18. Wood, H., Wood, D. Help Seeking, Learning, and Contingent Tutoring. *Computers and Education* vol.33 (1999) 153-159