Investigating Differential Error Types Between Human and Simulated Learners

Daniel Weitekamp, Zihuiwen Ye, Napol Rachatasumrit, Erik Harpstead(✉), and Kenneth Koedinger

Carnegie Mellon University, Pittsburgh, PA 15213, USA
harpstead@cmu.edu

Abstract. Simulated learners represent computational theories of human learning that can be used to evaluate educational technologies, provide practice opportunities for teachers, and advance our theoretical understanding of human learning. A key challenge in working with simulated learners is evaluating the accuracy of the simulation compared to the behavior of real human students. One way this evaluation is done is by comparing the error-rate learning curves from a population of human learners and a corresponding set of simulated learners. In this paper, we argue that this approach misses an opportunity to more accurately capture nuances in learning by treating all errors as the same. We present a simulated learner system, the Apprentice Learner (AL) Architecture, and use this more nuanced evaluation to demonstrate ways in which it does and does not explain and accurately predict student learning in terms of the reduction of different kinds of errors over time as it learns, as human students do, from an Intelligent Tutoring System (ITS).

Keywords: Simulated learners · Learning curves · Apprentice Learner

1 Introduction

Simulated learners are artificially intelligent agents that simulate human learning. Simulated learners offer a powerful set of affordances to AI powered instructional technology. Prior work has demonstrated the use of simulated learners for efficient authoring of intelligent tutoring systems [9,20], building automated learning by teaching exercises [10], and automated testing and refinement of educational technologies [5,18].

Simulated learners differ from parameterized statistical models like the Additive Factors, Performance Factors, and similar models [3,14] in that they fully simulate the process of human learning, not just the patterns of performance students exhibit over the course of learning. Simulated learners work in and learn from educational technology through an inductive process of skill creation and refinement. In this study we use simulated learners built with the Apprentice Learner Architecture, a modular framework for building simulated learners and testing computational theories of human learning [8]. Unlike deep learning

I. I. Bittencourt et al. (Eds.): AIED 2020, LNAI 12163, pp. 586–597, 2020.
https://doi.org/10.1007/978-3-030-52237-7_47

based simulated learners [16], Apprentice Learner (AL) agents reach mastery at roughly the same rate per opportunity as human learners, and make strong commitments to the theoretical underpinnings of learning without relying on highly parameterized fitting of human data [21].

To fully deliver on their potential to aid in the testing of instructional technology, simulated learners must embody an accurate theory of learning which can both reproduce the patterns of errors that humans produce over the course of learning, and respond as humans do in different instructional conditions. Prior work has assessed the fidelity of simulated learner models by comparing simulated student learning curves of error rate by opportunity to the learning curves of human learners. Macllelan [6] for example, presents simulated learners that shows similar learning curve patterns as humans trained under both blocked and interleaved instruction strategies. While this method has been helpful in guiding cognitive architectural decisions in the past, it has a potential to hide nuances in learners' behavior (e.g., doing a step incorrectly in different ways) that are also important for a simulation to model.

In this work, we demonstrate a method of assessing the accuracy of a simulated learner model not just by comparing overall learning curves, but also by a novel method of splitting learning curves by error type. While prior work has explored disaggregating learning curves by student subpopulations [11], our new method of generating learning curves draws two distinctions, first, between errors of *omission* whereby a learner's request for help is an indication that they do not know what do (Hint-Errors) and errors of *commission* where a student performs an incorrect action (Incorrects). Second, we make a distinction within Incorrects between actions on the wrong interface element, such as doing a step in the wrong order (Selection-Errors), and entering an incorrect value on an otherwise correct next step (Input-Errors). In the context of many tutoring systems, this distinction often appears as a difference between students putting any answer in an inappropriately selected text field (e.g., one they may use later on in the problem) and students putting an incorrect answer in an appropriately selected text field (e.g., making an arithmetic error). Tutoring systems commonly allow for multiple strategies, as is the case here, such that there may be multiple appropriate selections at some states in the solution.

An additional difficulty with modeling humans with current simulated learners [6,9], is that they use and acquire only domain-specific knowledge and, perhaps reasonably enough, they start with none. As such, they always begin learning with a 100% error rate. In principle, there is also a point at which human learners possess zero knowledge of a domain, however in the classroom setting, it is generally the case that most students have received at least some within-domain instruction prior to working with an intelligent tutoring system (ITS). Students also may possess some knowledge from previously learned domains that may sometimes provide correct solutions in the current domain of study. For the purposes of comparing the learning curves of humans and simulated learners, a comprehensive history of student learning is rarely available, and thus simulated learners must account for unobserved prior knowledge in their human counter-

parts. Weitekamp et al. [21], have compared several methods for accounting for prior knowledge in simulated learners. In addition to our error type analyses, we also incorporate several innovations on the best reported method of accounting for prior knowledge from this work.

Ultimately, we propose to improve learning theory by evaluating whether simulated learners that implement the computational theory of the Apprentice Learner Architecture and account for prior knowledge, can accurately predict (and thereby explain) the reduction in distinct types of errors produced by human learners; not just in overall error rate. Thus, we claim simulated learners capable of matching human learners' performance on all three of these error types (Hint-Error, Selection-Error, and Input-Error) constitute stronger models of human learning than those only capable of matching human learners on aggregate error-rates. Furthermore, we show that splitting the errors in these ways can help generate insights for how to refine simulated learner models and improve learning theory.

2 The Apprentice Learner Architecture

The simulated learners we employ throughout this work are implemented within a modular framework for generating simulated learners called the Apprentice Learner (AL) Architecture [6,8]. A single AL agent is a simulation of a single human learner, which learns as humans do through demonstrations and correctness feedback. AL agents can be trained interactively or, using an existing ITS. The Apprentice Learner Architecture's modular design consists of several independent learning mechanisms that can be swapped in and out to test different computational theories of human learning. Together, an AL agent's different learning mechanisms generate and refine production rules [1] that represent the skills of the agent. The left-hand side or if-part of each production rule is refined by *when-learning* and *where-learning* mechanisms, and the right-hand side or then-part of each production rule is generated by a *how-learning* mechanism.

In a typical AL agent, the *how-learning* mechanism is the first learning mechanism to come into play during learning. This mechanism induces a sequence of operations to explain how the action parameters (e.g. value) of a demonstrated training example were produced. *How-learning* searches over a set of domain-general operators such as addition, subtraction, multiplication, and division to find a sequence of operations that will constitute the then-part of a production rule capable of matching a demonstrated input. In this study we evaluate learning in fraction arithmetic, which only necessitates searching over singular unchained operators.

The *where-learning* mechanism is responsible for producing matching rules associated with each skill that can bind to the interface elements in an ITS interface associated with a particular use of a skill. For example, if a particular skill involves multiplying two numbers and placing the result in a text box, then the matching rule would need to bind to the text box (the selection) and to the two interface elements (the arguments) from which the solution will be

computed. If there are multiple instances of a step in a problem then the *where* matching rule would need to match to all such instances. In this work we employ a simple *where-learning* mechanism that simply recalls previously seen matches.

Lastly, the *when-learning* mechanism is responsible for learning when it is appropriate for a skill to be applied. *When-learning* mechanisms are simply binary classifiers, which take the current state of the problem and, for each skill, evaluate each *where* match in the state for that skill to determine whether or not the skill should be applied for that match. *When-learning* mechanisms generalize from correct and incorrect instances of a skill being applied to determine which features of a state indicate that a particular skill should be applied. In this study we test two different classification algorithms for *when-learning*, which have been used in prior studies with the Apprentice Learner [6] the Decision Tree algorithm a common classification algorithm [2], and TRESTLE which was used in prior work to model the gradual process of concept formation from examples [7].

3 Method

We evaluate our simulated learners against human data collected from a classroom study of a fraction arithmetic ITS. This dataset consists of the work of 117 students solving fraction addition and multiplication problems. Among the addition problems some problems involved adding fractions with the same denominator meaning the numerators could simply be added together, other problems involved adding fractions with different denominators, meaning a common denominator had to be found. For the later case, the ITS enforced the 'butterfly' method where a common denominator is found by multiplying the two denominators. All three problem types, Add-Different (AD), Add-Same (AS), and Multiply (M), were solved on the same interface. This dataset was used in prior work with simulated learners [6,21] and we have chosen to use it in this work for the sake of comparison. The dataset is available as project 243 on the PSLC DataShop [4][1].

In this study, we use a novel method of learning curve analysis that categorizes student errors into several different types. Following conventions from DataShop and ITS research [4,15], we frame student actions in terms of Selection-Action-Input (SAI) triples and consider errors along each dimension of the SAI. Selection is the interface element in the tutoring system that the student interacted with during an attempted step, Action is what they did to that interface element, and Input is the value associated with that action. For example, (num3, UpdateTextArea, 5) is the SAI for placing 5 in the textbox labelled num3.

Our method for defining error types leverages the behavior graph of a CTAT example-tracing tutor to annotate each student transaction with four new binary values by comparing a student's SAI against the problem step the transaction is associated with. These four new values are "current selection" which indicates whether a student worked on a correct selection for the next step, "current

[1] https://pslcdatashop.web.cmu.edu/Project?id=243.

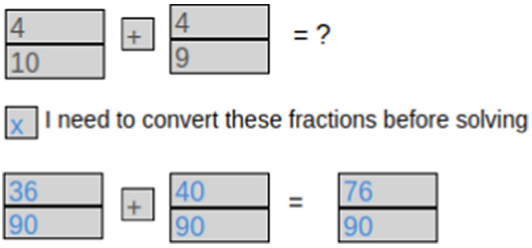

Fig. 1. An example of a fraction addition problem with different denominators (AD). The students must indicate that the fractions must be converted and apply the conversion using the "butterfly" method (multiplying the denominators together, and cross multiplying to find the new numerators). AS and M problems use the same interface but without the intermediate conversion steps.

input" which indicates if the input was correct, "downstream selection" which indicates whether the student's choice of selection is correct for a later step in the problem, and "downstream input" which indicates whether the input of the student's transaction would be correct on any step down stream in the behavior graph from the current step. Table 1 shows a few common patterns of these new values and how we group them together to get Selection-Errors and Input-Errors. Several combinations have been omitted because they are either impossible or not applicable to our data.

Table 1. Error types by SAI matching pattern

Current selection	Current input	Downstream selection	Downstream input	Error type
1	1	0	0	Correct response
1	0	0	0	Input-Error
0	0	1	0	Input-Error
0	1	1	0	Selection-Error
1	0	0	1	Selection-Error
0	0	1	1	Selection-Error
0	1	1	1	Selection-Error

In general, Selection-Errors occur whenever the student's current selection is wrong, but their input is applicable somewhere later in the problem, while Input-Errors occur when the student's input is incorrect for any step in the problem. Our motivation for encoding these distinct types of errors was to determine which characteristics of AL's learning mechanisms differed from human learners. Selection-Errors roughly correspond to issues of over-generality in the left-hand side of production rules (i.e. skills). For example, if a student does the wrong

step in a problem then that is an indication that they do not fully understand the conditions under which a particular skill should be applied. Input-Errors can arise when the right-hand side of a production rule is incorrect, however, they can also occur if a student applies the wrong skill for the correct next step, in which case, the Input-Error may arise from two or more skills with underspecific left-hand sides. In the context of AL, this means that Selection-Errors are definitely issues of under specific rules generated by the *when-* or *where-learning* mechanisms. Input-Errors, on the other hand, could arise from any learning mechanism, however, we hypothesize incorrect *how-learning* is most likely to show up as an Input-Error. Lastly, Hint-Errors can occur if no learning has occurred yet, or if *when-* or *where-learning* has generated skills with overly specific left-hand sides.

Before working with an ITS, human learners generally have some prior exposure to learning materials or instruction. However, when AL agents are first instantiated they have no such prior knowledge. Weitekamp et al. [21] attempted to estimate the number of prior practice opportunities per knowledge component needed to get a set of simulated learners on par with their human counterparts by extrapolating backwards with AFM [3]. In this work, we attempt to account for prior knowledge opportunities more precisely by using a pool of simulated learners trained on randomly generated problems. We estimate the number of prior opportunities per KC by finding the opportunity at which the pool of agents' learning curves best align with the first opportunity rate of the human data. To model each student individually we perturb the log odds of the target error rate by the AFM student intercept, yielding an individualized number of estimated prior opportunities for each knowledge component per student. This estimate is then used to pretrain each agent before it practices on the set of problems solved by its human counterpart.

Whereas [21] trained using only whole problems, we developed a new training procedure capable of training individual knowledge components. For all of the knowledge components of a particular problem type we train agents on random problems up to the minimum number of estimated prior opportunities over the constituent knowledge components. Any opportunities needed beyond this point are trained by having the agent solve problems from start to finish as usual, but only providing agents feedback on steps associated with knowledge components that still need practice. This new pretraining procedure can be applied to any step-based ITS with labelled KCs [17].

4 Results

Figure 2 shows learning curves for each type of error compared across the human data and AL using the two different *when-learning* mechanisms Decision Tree and TRESTLE. In accounting for prior knowledge, we find different results across the two methods. For the TRESTLE condition the overall error rate on the first opportunity is equivalent to the first opportunity error rate in the human data. For the Decision Tree, we find that AL has a first opportunity error rate that

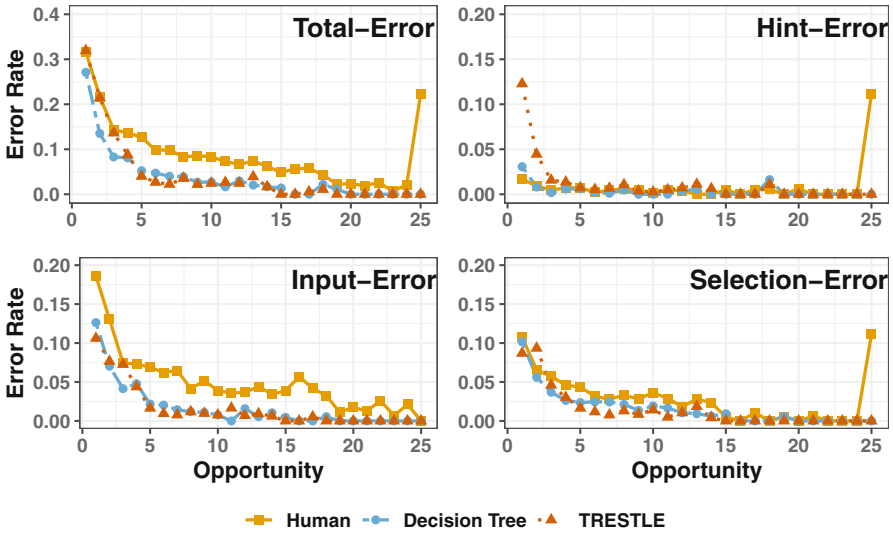

Fig. 2. Error-rate learning curves for each type of error (Total-Error, Hint-Error, Selection-Error, Input-Error) plotted by dataset (Human, AL with Decision Tree, AL with TRESTLE). Note the difference in y-axis scale between Total-Error and the others

is 4% lower than the human error rate – AL received on average too many pre-training opportunities and "over shot" the student state, indicating an imperfection in the pre-training method that we discuss below. In both conditions these results show an improvement over prior work which reported a first opportunity error rate discrepancy of 11% on the same dataset [21].

Overall, we find that AL agents with both *when-learning* methods learn more rapidly by opportunity than the human students. The AFM slope averaged over all KCs is 2.1 times greater than the humans for AL in both the Decision Tree and TRESTLE conditions. Additionally, In the Decision Tree and TRESTLE conditions the proportion of Input-Errors to Selection-Errors is roughly 1.1 whereas the human students make these errors at a ratio of 1.7. Both AL and the human students make almost all of their Hint-Errors within the first 5 opportunities. Only 5% of the human errors in the first 5 opportunities are Hint-Errors, however, TRESTLE based agents make considerably more Hint-Errors in these 5 opportunities at 26%. The Decision Tree makes 8% Hint-Errors over the first 5 opportunities, which is 60% more Hint-Errors than in the human data.

5 Discussion

We have compared AL agents with two different *when-learning* configurations to human data collected from a fraction arithmetic ITS. We have made this comparison using our novel method for splitting error curves by error type. In

this section we discuss the implications of these results toward converging on a more accurate computational model of human learning. We discuss potential future features of the Apprentice Learner Architecture, and discuss how these changes would affect the trends of AL agent learning curves split by error type.

5.1 Accounting for Prior Knowledge

The pretraining method outlined in Weitekamp et al. [21] pretrained simulated learners to within 11% of the human first opportunity rate. In this study, we have used the same TRESTLE *when-learning* strategy and dataset, but have employed a new method for accounting for prior knowledge in our simulated learners which matched the human first opportunity rate to within less than 0.1%. Our strategy was however, not quite as successful with the other agent configuration, which used the Decision Tree *when-learning* mechanism. This remaining discrepancy between AL and the human data may have to do with the granularity by which we can estimate prior opportunities for each knowledge component. The first opportunity error rate for some of the knowledge components in the human data is well over 50%, but for most knowledge components the pool of AL agents trained on random fraction arithmetic problems take only 4 or fewer opportunities on average to learn beyond 50% error. This means that our pre-training strategy is still fairly sensitive to rounding. When finding the best whole number of opportunities to pretrain to get the desired first opportunity error rate, rounding to the nearest opportunity can lead to large discrepancies since the difference in error rates between early opportunities is large.

5.2 Learning Curve Comparison

The split error curves in all three conditions indicate that our AL agents' pace of performance improvement by opportunity is much higher than the human students. Thus, we would expect any method for lowering the amount of learning per opportunity to improve the fit of our simulated learners to human data. Additionally, slowing down our agents' learning in this way would alleviate some of the rounding issues we have had with estimating prior knowledge since the learning curves would be less steep overall. The issue remains of determining how learning should be slowed. Recall that a simulated learner embodies a theory of human learning, thus our objective should not simply be to make our learners fit better to human data, but to do so in a theoretically and empirically grounded manner in order to converge on a model of human learning that is predictive across a wide range of domains and conditions.

On the empirical side, our learning curves split by error type provide a few insights concerning our three configurations of AL agents. Firstly, we find that the AL agents that employed TRESTLE made considerably more Hint-Errors than human students early in the learning process. For an AL agent a Hint-Error is committed when it encounters a problem state in which it believes that none of its learned skills are applicable. When this occurs, AL agents ask for a demonstration of the next correct step. Hint-Errors generally occur early on in learning

when skills either do not exist or have induced *when* and *where* conditions that are overly specific to previously seen training examples. In other words, Hint-Errors occur either when there is no appropriate *how* function induction for the then-part or when the preconditions for applying the correct skill for a step have not yet generalized to the point that they have become inclusive of all potential correct uses of that skill.

One capability which AL agents currently lack that may reduce the rate of Hint-Errors and increase the rate of Incorrect responses is the ability to make a plausible inference based on "weak methods" for more general problem solving [12] or by guessing, perhaps based on past response frequency. Human students sometimes rely on weak methods or guessing in the absence of strong hypotheses for what to do next [19]. Plausible inference may be informed by prior knowledge and involve actually taking actions similar to those in prior learned domains, or may even be slightly superstitious, (i.e., this seems like the kind of problem where the answer is 0) or based on interface heuristics (i.e., I usually operate on things that are next to each other). Further investigations are needed to select from or derive weak methods for plausible inference.

Another method for reducing Hint-Errors would be to use a *when-learning* mechanism that generalizes heavily from positive examples or incorporates negative feedback conservatively so that skills tend to be applied in spite of negative feedback. The Decision Tree appears to have this characteristic more so than TRESTLE.

One approach to plausible inference is to incorporating a memory mechanism. One weak method for plausible inference is to propose the action with the highest current memory activation (e.g., because of recency or history frequency of repetition or spacing). The AL agents tested in this work have no current means for such inference, and correspondingly, no means for forgetting skills or inferences. AL agents could incorporate methods of forgetting prior examples, features of problem states, induced internal states, or whole skills. While most existing literature on memory mechanisms pertains to the effects of memory on learning facts [1], it may be that a model like Anderson's ACT-R model of practice spacing and retention [13] is applicable to skills as well as facts. Overall a memory mechanism would likely slow down the learning rate of AL agents, although the effects of a memory mechanism on the proportion of Hint-Errors to Incorrect responses would likely depend on the implementation. The inclusion of a model of forgetting entire skills would likely further increase the number of Hint-Errors, however the spurious activation of other skills may make up for this and produce more Incorrect responses.

5.3 The Relative Rate of Input and Selection Errors

Another empirical result from our split learning curves is that among Incorrect responses human learners consistently make a larger proportion of Input-Errors than Selection-Errors over the course of learning. By contrast, AL agents consistently make these errors at about the same rate. One likely explanation for this difference is that the human students' Input-Error learning curve includes

instances of arithmetic mistakes when computing the right-hand side operations of skills. Currently, AL agents employ domain general operator functions to perform arithmetic and thus are incapable of making this kind of error. It may be possible to further split out these errors as a separate type of error with their own learning curve. One possible method for separating out these sorts of errors would be to use the methods employed by AL's *how-learning* mechanism in the error type labelling process to find errors which cannot be explained by applying weak methods on the values in the interface.

5.4 Other Uses of Learning Curve Splitting

Our method of splitting learning curves likely has uses for student modeling outside of the realm of simulated learners. Analyzing the rate of Selection-Errors and Input-Errors separately may help measure the efficacy of interventions baked into ITSs. For example, CTAT tutoring systems often correct students when they are working on the wrong step of a problem (i.e., a Selection-Error). Adding elaborative feedback to these messages to explain what the correct next step is and why it is correct may improve "if" and "then" type learning differently. The relative effect of such an intervention on these two types of learning could be measured directly with split learning curves to help refine feedback messages.

In this study we have grouped several distinct patterns into just two groups, but there may also be uses for splitting errors further. For example, one pattern we encode picks out cases where students have provided a correct answer for a later step. Analyzing the rate of this kind of error may help catch instances where a tutoring system arbitrarily constrains the order that steps can be taken. It may also help identify cases where students are restricted from providing a final answer produced through mental steps.

6 Conclusion

Just as theoretical physics complements experimental physics we suggest here, a need for more computational learning science to complement experimental learning science. Simulated learners are computational theories of human learning which model inductive human learning processes by working in and learning from ITSs. Evaluating and refining simulated learners as computational theories requires measuring the accuracy with which simulated learners match the specific learning behaviors of humans. In this work, we have presented two new methods to help make this comparison more precise. We have developed an improvement on previous methods [21] for accounting for prior knowledge in simulated learners, and we have developed a new method of splitting learning curves by error type.

We have employed these two methods in a comparison of simulated learners built with the Apprentice Learner Architecture and found that when prior knowledge is accounted for, AL agents learn about twice as fast as human learners, commit more initial Hint-Errors than humans, and produce a lower proportion

of Input-Errors to Selection-Errors. Finally, we have discussed several potential refinements of our current model based on these results such as alterations to *when-learning* mechanisms and the inclusion of mechanisms for forgetting, and the usage of weak methods that produce plausible inferences or guesses.

References

1. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychol. Rev. **111**(4), 1042–1044 (2004)
2. Breiman, L.: Classification and Regression Trees. Routledge, Boca Raton (2017)
3. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis – a general method for cognitive model evaluation and improvement. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 164–175. Springer, Heidelberg (2006). https://doi.org/10.1007/11774303_17
4. Koedinger, K.R., Baker, R.S., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A data repository for the EDM community: the PSLC datashop. In: Romero, C., Ventura, S., Pechenizkiy, M., Baker, R. (eds.) Handbook of Educational Data Mining, pp. 43–56. CRC Press, Boca Raton (2010)
5. Li, N., Cohen, W.W., Koedinger, K.R., Matsuda, N.: A machine learning approach for automatic student model discovery. In: EDM, pp. 31–40. ERIC (2011)
6. MacLellan, C.J.: Computational models of human learning: applications for tutor development, behavior prediction, and theory testing. Ph.D. thesis, Carnegie Mellon University (2017)
7. MacLellan, C.J., Harpstead, E., Aleven, V., Koedinger, K.R., et al.: Trestle: a model of concept formation in structured domains. Adv. Cogn. Syst. **4**, 131–150 (2016)
8. Maclellan, C.J., Harpstead, E., Patel, R., Koedinger, K.R.: The apprentice learner architecture: closing the loop between learning theory and educational data. In: International Conference on Educational Data Mining Society (2016)
9. Matsuda, N., Cohen, W.W., Koedinger, K.R.: Teaching the teacher: tutoring simstudent leads to more effective cognitive tutor authoring. Int. J. Artif. Intell. Educ. **25**(1), 1–34 (2015)
10. Matsuda, N., Keiser, V., Raizada, R., Stylianides, G., Cohen, W.W., Koedinger, K.R.: Learning by teaching simstudent – interactive event. In: Biswas, G., Bull, S., Kay, J., Mitrovic, A. (eds.) AIED 2011. LNCS (LNAI), vol. 6738, pp. 623–623. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21869-9_124
11. Murray, R.C., et al.: Revealing the learning in learning curves. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 473–482. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_48
12. Newell, A., Shaw, J.C., Simon, H.A.: Report on a general problem solving program. In: IFIP Congress, Pittsburgh, PA, vol. 256, p. 64 (1959)
13. Pavlik, P.I., Anderson, J.R.: Using a model to compute the optimal schedule of practice. J. Exp. Psychol. Appl. **14**(2), 101 (2008)
14. Pavlik Jr, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis - a new alternative to knowledge tracing. Online Submission (2009)
15. Ritter, S., Anderson, J.R., Koedinger, K.R., Corbett, A.: Cognitive tutor: applied research in mathematics education. Psychon. Bull. Rev. **14**(2), 249–255 (2007)

16. Roads, B.D., Mozer, M.C.: Predicting the ease of human category learning using radial basis function networks (2019)
17. VanLehn, K.: The interaction plateau: answer-based tutoring step-based tutoring = natural tutoring. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoje, S. (eds.) ITS 2008. LNCS, vol. 5091, p. 7. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69132-7_4
18. VanLehn, K., Ohlsson, S., Nason, R.: Applications of simulated students: an exploration. J. Artif. Intell. Educ. **5**(2), 1–42 (1994)
19. Waller, M.I.: Modeling guessing behavior: a comparison of two IRT models. Appl. Psychol. Meas. **13**(3), 233–243 (1989)
20. Weitekamp, D., Harpstead, E., Koedinger, K.: An interaction design for machine teaching to develop AI tutors. In: CHI (2020)
21. Weitekamp III, D., Harpstead, E., MacLellan, C.J., Rachatasumrit, N., Koedinger, K.R.: Toward near zero-parameter prediction using a computational model of student learning, Ann Arbor (2009)