Toward Stable Asymptotic Learning with Simulated Learners

Daniel Weitekamp(✉), Erik Harpstead, and Kenneth Koedinger

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

Abstract. Simulations of human learning have shown potential for supporting ITS authoring and testing, in addition to other use cases. To date, simulated learner technologies have often failed to robustly achieve perfect performance with considerable training. In this work we identify an impediment to producing perfect asymptotic learning performance in simulated learners and introduce one significant improvement to the Apprentice Learner Framework to this end.

Keywords: Simulated learners · Cognitive modeling · Authoring tools

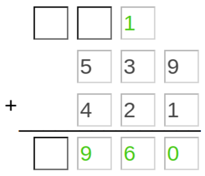
1 Introduction

Simulated learners are simulations of human learning that learn to perform tasks through an interactive process of demonstrations and feedback provided either by a human tutor or an Intelligent Tutoring System (ITS). Simulated learners have the potential to revolutionize learning technologies on a number of fronts, Matsuda demonstrated that students can learn by teaching a simulated learner called SimStudent [11], and Li showed the potential of SimStudent for cognitive model discovery [5]. Additionally, Matsuda, Maclellan, and Weitekamp [7,10,13] have demonstrated the use of simulated learners as a potential means of authoring ITSs [12], such as cognitive tutors [4], more efficiently than comparable methods [2] that do not employ simulated learners.

For the purposes of using simulated learners as authoring tools it is desirable that the performance of the simulated learner asymptotically tends toward zero error. This capability ensures that the ultimate tutoring system behavior learned by the agent does not mark correct student responses as incorrect or incorrect responses as correct. We explore the asymptotic performance of simulated learners using the Apprentice Learner (AL), a modular software library for creating simulated learners instantiating different mechanistic theories of learning [6]. In this work, we identify a new source of learning failure that prevent AL agents from achieving zero training error and demonstrate an adjustment to the AL framework that allows it to recover from this failure mode. More broadly this work identifies and remedies an issue unexplored by prior inductive task learning literature, how an agent can recover from an incorrect induction made early in training to asymptotically acquire a knowledge state functionally equivalent to a set of ground-truth procedural knowledge.

2 Training Test Domain: Multi-column Addition ITS

To demonstrate issues in aysmptotic training behavior, we use mutli-column addition as a simple prototypical example. We train our agents on an ITS implemented with CTAT's [1] nools [9] model tracer [3] that supports practice on what is often called the "standard" or "traditional" algorithm for adding large numbers, whereby the digits of the numbers to be summed are aligned in columns, summed column by column, with an extra "carry" row that is used for carrying the tens' digit from one column to the next.



3 A Brief Overview of the Apprentice Learner Framework

Apprentice Learner agents learn a set of skills sufficient to apply target tasks by learning in an interactive process with an ITS or human author. Each skills that an AL agent learns has at least four parts *how*, *where*, *when*, and *which*, that are each learned by different learning mechanisms. *How-learning* learns the *how-part* a composition of domain general functions that produces an action. For example, after trying a number of operations in different combinations *how-learning* might learn the *how-part* Mod10(Add(?.v, ?.v)) which takes two interface elements (the ?s) as arguments, sums their values (the .v's) and takes their one's digit (i.e. the modulus of 10).

A found RHS can work for a particular example but fail to work in general, for example another explanation an AL agent may come up with from the previous example is Copy(?.v) or just copy the value of the second value in a column into the carry slot. For 539+421 this would work for the first carry step, but would fail in general. In the AL agents used in this study a skill is identified by its RHS, so if the RHS happens to be wrong a new one must be induced, and the old one must be overridden or discarded.

Where-learning learns the *where-part*—a set of rules that pick out a set of arguments for a RHS, for example all of the numbers above the line in a column, and a "selection", the field into which the evaluation of the RHS will be placed. *When-learning* learns the *when-part* conditions, over the whole state, under which a skill should fire given a proposed *where-part* binding. Finally *which-learning* learns a policy for picking which potential application of a skill should be applied if multiple pass the *where-* and *when-part* rules. Given space constraints the reader should refer to prior work for further details about these learning mechanisms [6,14].

4 Addressing Lingering Weak and Overgeneral Skills

AL agents may need to observe several examples of taking particular problem steps to induce the correct *how-part* for the true skill associated with that kind of step. In the meantime a weak (i.e. not correct in all situations) *how-part* can be induced. Skills with weak *how-parts* will tend to be buried by building up a low *which-part* utility through repeated negative feedback, whereas correct skills may accumulate some negative feedback as their *when-part* conditions are refined and fewer later on. Consequently correct skills tend to override weak skills by accumulating a higher *which* utility. Prior work with simulated learners has shown that overriding via *which* utility works well in many domains [8], but we have identified some circumstances that necessitate a revaluation of this method.

For instance, it is possible for *how-parts* to be misattributed to the wrong skill. Consider for example, the case of an untrained simulated learner seeing $215 + 846$, and asking for examples of how to do the first few steps. Adding the 5 and 6 produces 11, creating an opportunity for the same skill to be induced and attributed to the first two actions (which should utilize seperate skills)—placing a 1 below and carry a 1 to the next column. Since the interface elements on which the two actions act are different, the *where-learning* mechanism for that skill will over-generalize the conditions constraining legal bindings of the selection field causing the agent to apply the skill in a number of absurd ways.

4.1 Two Methods for Addressing Overgeneralization Errors

To address overgeneralization issues in AL agents we present two possible implementation changes and evaluate each independently. First, we implement a means for faulty skills to be removed including those that have overgeneralized. Second, we implement a *where-learning* mechanism that is capable of undoing generalization errors. A key observation in both proposed implementation changes is that faulty skills, either those with incorrect RHSs or overgeneralized where-part rules, will tend to make more errors than non-faulty ones, especially late in the training process. From a cognitive standpoint these can be thought of as persistent weak hypotheses of the true procedure. But these weak hypotheses should not persist indefinitely in the face of negative reinforcement, and should eventually be given up on.

Our first implementation change is to add a new removal utility, a number between 0 and 1 that when lowered below a threshold of .2 signals that a skill should be removed from an agent. We try three different functions of "p" and "n" (the numbers of instances of positive and negative feedback) for this utility: 1) the proportion correct $p/(p+n)$ (same as the *which* utility) 2) double counted negatives $p/(p+2n)$, and 3) nonlinearly counted negatives $p/p+n+1/4n^2)$. Nonlinearly counted negatives implements the intuition that skills that persistently produce errors after considerable training are more likely to be faulty than skills that only produce errors initially while a skills *when-part* rules are still being refined.

Our second implementation change introduces a fourth condition called "recovering where" that enables overgeneralized *where-part* conditions to return to a more specific state. Each newly generalized set of *where-part* conditions has its own removal utility that is updated, when applicable, with positive or negative feedback, and is removed when the utility calculated on the counts of positive and negative feedback falls below a threshold of .5.

4.2 Results of Implementation Changes

For all tested variations of the two proposed implementation changes we ran 100 agents on 100 3 $\times$ 3 multi-column addition problems. The first problem is always fixed to $215 + 846$ to ensure that a large number of the agents exhibit the *where-part* overgeneralization issue, and the remaining 99 problems are sampled randomly.

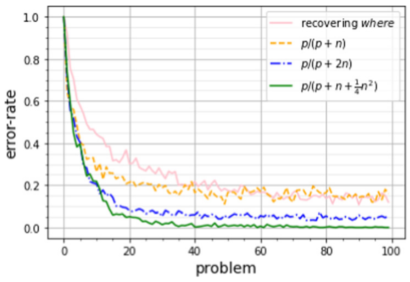


Fig. 1. Comparison of four recovery methods from *where-part* overgeneralization

Among the implementations of skill removal utility, nonlinearly counted negatives (i.e. "$p/(p+n+1/4n^2)$") reliably removed overgeneralized and persistent weak skills, while the other methods still exhibited persistant error. We suspect that the 'recovering where' condition was ineffective because each competing *where-part* generalization shares a *when-learning* mechanism, meaning that even if bad generalizations are eliminated, eventually a considerable number of unusual training instances centered around irrelevant selection fields will remain in the *when* training history, making it far more challenging to establish a set of consistent *when-part* conditions. Whole skill removal by contrast is a consistently more effective method of over-generalization removal since removing an entire skill allows for a new skill to be induced in its place, giving *when-learning* a clean slate to work with.

5 Conclusion

In this work we have identified challenges to achieving asymptotic performance with simulated learners and remediated sources of persistent asymptotic error in simulated learners implemented with the Apprentice Learner Framework.

References

1. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 61–70. Springer, Heidelberg (2006). https://doi.org/10.1007/11774303_7
2. Aleven, V., Sewall, J., McLaren, B.M., Koedinger, K.R.: Rapid authoring of intelligent tutors for real-world and experimental use. In: Sixth IEEE International Conference on Advanced Learning Technologies (ICALT 2006), pp. 847–851. IEEE (2006)
3. Blessing, S.B., Gilbert, S.B., Ourada, S., Ritter, S.: Authoring model-tracing cognitive tutors. Int. J. Artif. Intell. Educ **19**(2), 189–210 (2009)
4. Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city (1997)
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., Harpstead, E., Patel, R., Koedinger, K.R.: The apprentice learner architecture: closing the loop between learning theory and educational data. In: International Education Data Mining Society (2016)
7. MacLellan, C.J., et al.: Authoring tutors with complex solutions: a comparative analysis of example tracing and simstudent. In: The 2nd AIED Workshop on Simulated Learners. CEUR-WS.org, Madrid, Spain (2015)
8. MacLellan, C.J., Koedinger, K.R.: Domain-general tutor authoring with apprentice learner models. Int. J. Artif. Intell. Educ. 1–42 (2020). https://link.springer.com/journal/40593/topicalCollection/AC_d47d1642e3402a9a8134c35afb7851c8
9. Martin, D.: Nools. https://github.com/noolsjs/nools
10. 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)
11. 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, p. 623. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21869-9_124
12. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educ. Psychol. **46**(4), 197–221 (2011)
13. Weitekamp, D., Harpstead, E., Koedinger, K.: An interaction design for machine teaching to develop AI tutors. In: CHI (2020). in press
14. Weitekamp, D., Ye, Z., Rachatasumrit, N., Harpstead, E., Koedinger, K.: Investigating differential error types between human and simulated learners. In: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E. (eds.) AIED 2020. LNCS (LNAI), vol. 12163, pp. 586–597. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52237-7_47