An Improved Data-Driven Hint Selection Algorithm for Probability Tutors

Thomas W. Price
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
twprice@ncsu.edu

Collin F. Lynch
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
cflynch@ncsu.edu

Tiffany Barnes
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
tmbarnes@ncsu.edu

Min Chi
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
mchi@ncsu.edu

ABSTRACT

Data-driven systems such as the Hint Factory have been successful at providing student guidance by extracting procedural hints from prior user data. However, when only small amounts of data are available, it may be unable to do so. We present a novel hint-selection algorithm for coherent derivational domains, such as probability, which addresses this problem by searching a frontier of viable, partially matching student states. We tested this algorithm on a dataset collected from two probability tutors and performed a cold start comparison with direct state matching. We found that our algorithm provided higher value hints to students in unknown states 55.0% of the time. For some problems, it also provided higher value hints in known states.

1. INTRODUCTION

Adaptive feedback is one of the hallmarks of an Intelligent Tutoring System. This feedback often takes the form of hints, pointing a student to the next step in solving a problem. While hints can be authored by experts, more recent data-driven approaches, such as the Hint Factory [1] have shown that this feedback can be automatically generated from prior student data. The Hint Factory operates on a representation of a problem-specific dataset called an interaction network [3], where each vertex represents the state of a student's solution at some point during the problem solving process, and each edge represents a student's action. A complete solution is represented as a path from the initial state to a goal state. A new student requesting a hint is matched to a previously observed state and directed along a path to the goal state.

If too few students have been recorded, the Hint Factory is unable to match new students to existing states in the network. This is known as the *cold start problem*, a fundamental challenge in many domains. For example, when Hint Factory's original state matching algorithm was applied to BOTS, an educational programming game, a dataset of nearly 100 students provided only 40% hint coverage [4].

This paper focuses on two probability tutors in which many actions have no ordering constrains. This can produce an exponentially large state space, making the cold start problem even harder to overcome. We present a novel state matching mechanism that helps address this problem in *coherent derivational domains*. These are problem-solving domains, such as probability, physics, and logic, where: *a*) a solution S is constructed by repeated applications of domain rules to derive a goal value; *b*) taking any valid action cannot prevent the student from taking another valid action; and *c*) if S is a complete solution to the problem, then any superset of S is also a complete solution. Note that this does not prevent rule applications within a solution from having ordering constraints.

2. SELECTION ALGORITHMS

For our purposes, we assume a hint selection algorithm takes the following inputs: *a*) an interaction network, $N = (V,E)$ of previously observed states and actions; *b*) a value or ordering function $f : V \to \mathbb{R}$, which assigns "desirability" to each of the states in V; and *c*) the current state s_c of a student who is requesting a hint. In coherent derivational domains, each state $s \in V$ can be defined by the set of derived facts. Each edge $e \in E$ is annotated with an action a_e, the derivation or deletion of a fact.

Given this information, a selection algorithm attempts to find the optimal action a, such that a is a valid action in state s_c, and the value of the resulting state $f(s_a)$ is maximized. Here we derive f from the Hint Factory's value iteration procedure [1], but other functions could be used instead.

The selection algorithm employed by the Hint Factory requires that $s_c \in V$, meaning the student is in a known, or previously observed state. The algorithm then selects the successor of s_c with the highest value and returns the action which leads to this state.

In the case that s_c is unknown, meaning $s_c \notin V$, Barnes and Stamper [1] suggest using a student's previous state to generate a hint. This approach can be generalized to walking back to the last recognized state in the student's path, and using that to generate a hint. We refer to this as the "Backup Selection" algorithm.

In our selection algorithm, we first mark all $v \in V$ such that $v \subseteq s$. Beginning with the start state s_0, we traverse the graph in a depth-first fashion, following an edge e only if a_e is a deletion or derives a fact which is present in s_c. Let us call the set of states traversed in the manner T. Note that we do not *generate* states here, but explore only the previously observed states in N. We know that for any $t \in T$, $t \subseteq s_c$ and t is reachable by a known path from the start state. We define the Frontier F as the set of all states which can be reached by a single action from a state in T. A student in s_c can reach any state in the Frontier – or some superset of the Frontier state – in a single action. We then find the edge $\overrightarrow{tu}$ which maximizes $f(u)$ and return its annotated action.

3. EVALUATION

Our evaluation was based on the cold start experiment originally used to evaluate the Hint Factory [1], which was designed to measure how much data was required to provide hints to new students. Because we can always provide *some* hint by applying the Backup algorithm, we are instead interested in measuring the quality of the hints being given. Since we cannot directly measure hint quality, we will use the value function, f, described in Section 2, as an approximation of the quality. Here we use the value iteration method employed by the Hint Factory [1]. We do not make the claim this is an ideal metric, and this experiment can be easily adapted to work with any value function.

We evaluated our algorithm using combined log data from the Andes and Pyrenees probability tutors [2]. The Andes data was drawn from a prior experiment [2] and included 394 problem attempts by 66 students over 11 problems. The Pyrenees data included 999 problem attempts by 137 students on the same problem set. The tutors contain the same knowledge base, problems and solutions, allowing their data to be merged. This allowed us access to a wider variety of data than a single tutor would afford.

3.1 Procedure

For each problem, a student was selected at random and removed from the population to represent a previously unobserved student. We will call this student's path P. The remaining students who successfully solved the problem were added, one at a time and in a random order, to the network, N. Let n be the number of students added this way. After each addition, for each non-solution state in P, we calculated hints with the Backup selection algorithm and with our algorithm. We gave each of these hints a value, equal to $f(s)$, where s is the resulting state of applying the hint. In the case that this state was not in N, we used the value of the Fringe state selected by the algorithm (a superset of the resulting state). If our algorithm showed an improvement, we also recorded whether or not the state requesting a hint was known, meaning it was in N. This process was repeated 500 times to account for ordering effects.

3.2 Results

For each problem, we averaged the the percentage of *unknown* states with improved hints over all values of n. This average ranges from 33.5% to 69.8%, with an average of 55.0%. This indicates that our algorithm accomplishes its intended purpose of improving hint selection when insufficient data makes it difficult to find matching states in the network. However, while we were able to improve hints for a large *percentage* of these unknown states, the number of unknown states dropped off rapidly as n increased.

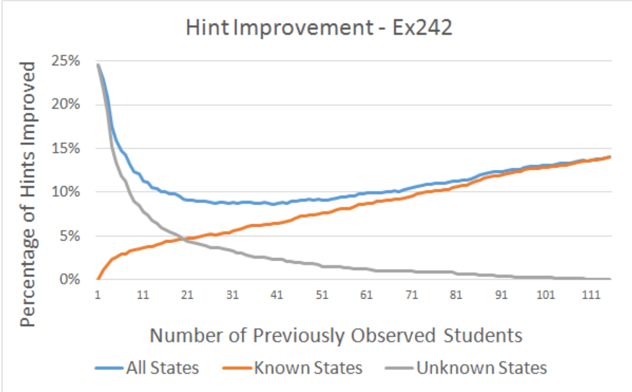

Figure 1: One cold start curve, showing the percent of hints which are improved by our algorithm (y-axis), given the number of students in N (x-axis).

For 7 of the 11 problems, our algorithm also produced improved hints for *known* states. Notably, the percentage of improved hints *increases* as more students are added to N, meaning additional data strengthens our algorithm's advantage. After all of the students were added to N, this number ranged from 3.6% to 49.7%, with an average of 17.8%. The improvement for known states seems to depend largely on the graph structure, and occurs infrequently in smaller graphs. Figure 1 depicts one cold start graph demonstrating the trends for known and unknown states.

4. CONCLUSIONS

We have presented a novel algorithm for selecting among possible data-driven hints. We have demonstrated that on average our algorithm gives a higher value hint 55.0% of the time when a student is in an unknown state, and 17.8% of the time for known states in a subset of problems.

5. ACKNOWLEDGMENTS

Work supported by NSF Grant #1432156 "Educational Data Mining for Individualized Instruction in STEM Learning Environments" Min Chi & Tiffany Barnes, Co-PIs.

6. REFERENCES

[1] T. Barnes and J. Stamper. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Intelligent Tutoring Systems (ITS)*, pages 373–382, 2008.
[2] M. Chi and K. VanLehn. Eliminating the gap between the high and low students through meta-cognitive strategy instruction. In *Intelligent Tutoring Systems (ITS)*, volume 5091, pages 603–613, 2008.
[3] M. Eagle and T. Barnes. Exploring Networks of Problem-Solving Interactions. In *Learning Analytics (LAK)*, 2015.
[4] B. Peddycord III, A. Hicks, and T. Barnes. Generating Hints for Programming Problems Using Intermediate Output. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pages 92–98, 2014.