The Validity of Providing Automated Hints in an ITS Using a MDP

John C. Stamper, Tiffany Barnes

Department of Computer Science
University of North Carolina at Charlotte
9201 University City Blvd., Charlotte, NC 28223
{jcstampe, tbarnes2}@uncc.edu

Introduction

In building intelligent tutoring systems, it is critical to be able to understand and diagnose student responses in interactive problem solving. However, building this understanding into the tutor is a time-intensive process usually conducted by subject experts. Much of this time is spent in building production rules that model all the ways a student might solve a problem. In our prior work, we have proposed a novel application of Markov decision processes (MDPs), to automatically generate hints for an intelligent tutor that learns. We demonstrate the feasibility of this approach by extracting MDPs from four semesters of student solutions in a logic proof tutor, and calculating the probability that we may be able to generate hints for students. Our results indicate that extracted MDPs will be able to provide over 80% of students with hints while working problems.

Background and Related Work

Through adaptation to individual learners, intelligent tutoring systems (ITS) can have significant effects on learning (Anderson 2001). However, building one hour of adaptive instruction takes between 100-1000 hours of work of subject experts, instructional designers, and programmers (Murray 1999), and a large part of this time is used in developing production rules that are used to model student behavior and progress. A variety of approaches have been used to reduce the development time for ITSs, including ITS authoring tools or building constraint-based student models instead of production rule systems. Constraint-based tutors, which look for violations of problem constraints, require less time to construct and have been favorably compared to cognitive tutors, particularly for problems that may not be heavily procedural (Mitrovic 2003). Some systems, including RIDES and DIAG use teacher-authored or demonstrated examples to develop ITS production rules (Murray 1999). These systems cannot be easily generalized, however, to learn from student data. CTAT has been used to develop "pseudo-tutors" for subjects including genetics, Java, and truth tables (Koedinger 2004). This system has also been used with data to build initial models for an ITS, in an approach called Bootstrapping Novice Data (BND) (McLaren 2004).

Similar to the goal of BND, we seek to use student data to directly create student models for an ITS. However, instead of feeding student behavior data into CTAT to build a production rule system, we proposed to generate Markov Decision Processes that represent all student approaches to a particular problem, and use these MDPs directly to generate feedback. We believe one of the most important contributions of this work is the ability generating feedback based on frequent, low-error student solutions. This method of using previous student data reduces the expert knowledge needed to generate intelligent, context-dependent feedback. The system we proposed is capable of continued refinement as new data is provided. We illustrated our approach by applying MDPs to analyze student work in solving formal logic proofs (Barnes 2007), and demonstrated the applicability of using MDPs to collect and model student behavior and generate a graph of student responses that can be used as the basis for an ITS. We performed a pilot study to extract rules from a MDP for a simple proof from three semesters of student data from a logic tutor called Deep Thought (Croy 2000). We verified that the rules extracted by the MDP conformed to expert-derived rules although a number of generated buggy rules surprised experts (Stamper 2006).

Experiment and Results

In our method, a tree is created for a particular problem with the states consisting of the group of premises, connected by the actions taken on those premises. Starting with each student attempt the state is the list of premises that the student currently has in the proof. This means for a given problem all students start at the same state. Also successfully solving the problem arrives at the same goal state. The list of premises can be ordered or un-ordered. An ordered list takes the order steps are performed into account while un-ordered does not. A tree is built using many student attempts and creates probabilities of taking actions based on the percentage of times these actions are

seen at a given state by the actual student attempts. Then using value iteration, values are assigned to each state. We start by assigning the goal state a high value (100 in our trials) and error states a negative number as a penalty (-10) in our case. All other states are given an initial value of zero. A transition cost of -1 is applied. After running the value iteration algorithm, the result is a MDP where each state has a value which describes how "good" the state is in relation to achieving the goal. We can give help in the form of hints from a given state by examining the action leading to the state with the best value from a given state. The hint is composed of the action, the resulting state, or a combination of both.

For this experiment, we used data from four fall semesters (2003-2006) of a discrete math course at NC State University. The number of students in each semester was 240, 263, 262, and 128. Students in this course are typically engineering and computer science students in their second or third year of college, but most have not been exposed to a course in logic. Students attend several lectures on propositional logic and complete an online homework where students complete truth tables and fill in the blanks in partially completed proofs. Students then use the Proofs Tutorial to solve 10 proofs, directly or using proof by contradiction. Fifty-eight percent of students used direct proof when solving proof 1. We extracted 523 of students' first attempts at direct solutions to proof 1 from the Proofs Tutorial.

Our hypothesis was that the method would ramp up quickly on the ability to give hints. The experiment setup follows. First, take a group of student attempts. Next, randomly pick one and use it to seed the state list. Then, from the remaining group randomly pick another attempt and see how many states in the attempt are available in the state list. These are the states we can give hints. Finally, add the new states to the state list and repeat for all of the remaining attempts. Repeat this procedure many times and the result will be the percent average of hint availability. The results summarized in table 1, show the experiment run on the group of 523 students for problem 1 in the NCSU dataset. The procedure was repeated 100,000 times to smooth the graph. Clearly, the availability to give hints ramps up very quickly. For un-ordered states (states where the order premises are created does not matter) the 50% threshold is reached at just 8 student attempts and the 75% threshold at 49 attempts. For ordered (where the sequence of creation matters), 50% occurs on attempt 11 and 75% on attempt 88.

Hint Percentage	50%	75%	85%	90%
Un-Ordered	8	46	154	360
Ordered	11	85	362	N/A

Table 1 – Hint Availability

Furthermore, the data gives evidence that an instructor could seed the data to jump start new problems. By allowing the instructor to enter as few as 8-11 example solutions to a problem, the method would already be capable of automatically generating hints for 50% student states.

Conclusions and Future Work

This work has verified that our method of using a MDP to automatically generate hints will cover the most common student approached with a very few number of student problem attempts. The next step is to integrate the approach into the actual computer based training. Once this is complete, experiments to determine the usage and effectiveness of the generated hints will be evaluated.

References

Anderson, J., Gluck, K. 2001. What role do cognitive architectures play in intelligent tutoring systems? *In D. Klahr & S. Carver (Eds.) Cognition & Instruction: 25 years of progress*, 227-262. Erlbaum.

Barnes, T., Stamper, J. (2007). Toward the Extraction of Production Rules for Solving Logic Proofs. *Papers from the 2007 AIED Educational Data Mining Workshop, ed. Joseph E. Beck.*, Marina Del Rey, CA, July 9, 2007.

Croy, M. 2000. "Problem Solving, Working Backwards, and Graphic Proof Representation," *Teaching Philosophy*, 23(2), 169-187.

Koedinger, K. R., Aleven, V., Heffernan. T., McLaren, B. & Hockenberry, M. 2004. Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. *Proc. 7th Intelligent Tutoring Systems Conference*, Maceio, Brazil, pp. 162-173.

Murray, Tom. 1999. Authoring intelligent tutoring systems: An analysis of the state of the art. *Intl. J. Artificial Intelligence in Education*, 10: 98-129.

Mitrovic, A., Koedinger, K. & Martin, B. 2003. A comparative analysis of cognitive tutoring and constraint-based modeling. *User Modeling* 2003: 313-322.

McLaren, B., Koedinger, K., Schneider, M., Harrer, A., & Bollen, L. 2004. Bootstrapping Novice Data: Semi-automated tutor authoring using student log files, In *Proc. Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, 7th Intl. Conf. Intelligent Tutoring Systems (ITS-2004)*, Maceió, Brazil, August 30, 2004.

Stamper. J. 2006. Automating the Generation of Production Rules for Intelligent Tutoring Systems. *Proc. Intl. Conf. Computer-Aided Learning (ICL2006).* Austria, Sep. 27-29, 2006.