Decision Support for an Adversarial Game Environment Using Automatic Hint Generation

Steven Moore(✉) and John Stamper

HCII, Carnegie Mellon University, Pittsburgh, USA
StevenJamesMoore@gmail.com, jstamper@cs.cmu.edu

Abstract. The Hint Factory is a method of automatic hint generation that has been used to augment hints in a number of educational systems. Although the previous implementations were done in domains with largely deterministic environments, the methods are inherently useful in stochastic environments with uncertainty. In this work, we explore the game Connect Four as a simple domain to give decision support under uncertainty. We speculate how the implementation created could be extended to other domains including simulated learning environments and advanced navigational tasks.

Keywords: Hint generation · Educational data mining · Reinforcement learning

1 Introduction and Related Work

Adaptive learning through the use of intelligent tutoring systems (ITS) have been shown to increase overall learning and decrease the time needed to mastery (Koedinger et al. 2013). To overcome the difficulty of ITS creation, developers have turned to authoring tools, like CTAT (Aleven et al. 2006) or data driven techniques (Stamper et al. 2007). While large educational data repositories like DataShop (Stamper et al. 2010) exist for many traditional educational systems, one area that has been less explored is decision support systems in domains that include multiple interacting students (or players or agents). These domains are significantly more complex because, unlike a traditional educational system, a specific action in these domains by a student may not have a deterministic outcome. In this work, we explore how a technique called the Hint Factory (Stamper et al. 2008) can be used for decision support (hints and feedback) in adversarial domains and specifically implement the technique using a Connect Four data set.

Previous work has utilized datasets from solved two-player games, such as Connect Four, often for reinforcement learning (RL) and temporal difference learning (TDL) tasks. One such study found that using a self-learning agent operating with the Minimax algorithm required 1.5 million games to establish an 80% success rate (Thill et al. 2012). We look to show how well the Hint Factory can be utilized in such domains, like adversarial games, in order to provide decision support in the form of hints and feedback.

A. Coy et al. (Eds.): ITS 2019, LNCS 11528, pp. 82–88, 2019.
https://doi.org/10.1007/978-3-030-22244-4_11

The Hint Factory provides context specific hints to students using a novel technique that creates graphs of past student solutions from student log data, which can then be used to suggest the best step for a student to solve the problem based on their current state in the graph. The Hint Factory, applied in a tutor for teaching deductive logic proofs, has been shown to significantly increase learning and reduce attrition (Stamper et al. 2011). Determining the timing and frequency of hints is a particular challenge, but studies suggest that offering hints on demand, instead of proactively, can have positive effects on learning (Razzaq and Heffernan 2010). While some studies have suggested as much as 72% of help-seeking behaviors can be unproductive (Aleven et al. 2004), Shih's work suggests that some of these behaviors are in fact helpful. They argue that using help to achieve a bottom-out hint can be seen as looking for a worked example, an effective learning strategy (Shih et al. 2011). On-demand, context-specific Hint Factory hints and feedback are effective in single user educational technologies and have been shown to significantly help students complete more problems resulting in higher outcomes on post tests (Stamper et al. 2011).

In Connect Four, there are three possible states for each of the forty-two available game spaces, with a game board configuration consisting of six rows and seven columns. The space can be occupied by the turn players piece, the opponent's piece, or it can be empty. Early work by Allis (1988) found that after ruling out possible illegal configurations that were present in the initial estimates, the possible board configurations were close to an upper bound of $7.1 * 10^{13}$. More recently, it was determined using binary decision diagrams that there are exactly 4,531,985,219,092 legal board configurations (Edelkamp and Kissmann 2008).

A key feature of the Hint Factory is the use of a Markov Decision Process (MDP) to create a policy based on data from previous attempts at problems in an educational system (Stamper 2006). The resulting policy allows the Hint Factory to determine the next best state and action a student can take. The features of the state and action are used to generate hints and feedback for the student or user (Barnes et al. 2011). In this work, states are represented by the current configuration of the Connect Four board at a given time step and a time step is the state of the board after both players have made their move. Actions cause the current state to transition into a new state, for this game there are at most seven possible actions represented by the column a game piece can be dropped into. Transitions, which have an associated probability, is the process of entering a specific new state, based on the current state and the player's chosen action. In Connect Four, a given state-action pair can result in at most seven new states, without accounting for the following adversarial agent response, which also moves into one of the seven available columns. This means that at most one state can lead into forty-nine following states accounting for both players' actions.

Rewards are a numerical value tied to entering a new state, we assign a high value for a winning endgame state, and a high negative value for a losing endgame state, and 0 for an endgame state resulting in a draw. All other game states begin with a value of 0 and then value iteration is applied to create a policy for the MDP (Sutton and Barto 1998). Our implementation utilizes a greedy policy, which prioritizes being on a path toward any winning state, regardless of how many turns it took. Finally, the value is the expected reward starting from a state, taking an action, and then following the given policy in place.

2 Implementation

To collect the Connect Four data, we used a web-based implementation with the logic of the game written in TypeScript (Anggara 2018). The program puts two computer players against one another to simulate the gameplay, recording each of their moves. In total we collected five thousand games played between the two computer players.

Both computer players used the same logic to play the game, which is the Minimax algorithm with alpha-beta pruning. The Minimax algorithm looks at all possible states, determines the worst possible outcome, and then selects a possible state that has the least worst outcomes (Edwards 1954). One non-standard modification we did to the algorithm is that the first move for each computer player is randomized. We found that causing each player's first move to randomly be placed in one of the seven columns created a greater diversity of games that closely resembled gameplay between two human players. Additionally, Minimax uses a heuristic function that weighs a reward in the closer future worth more than the same reward in the distant future. This means it favors moves that win the next turn, prevent the opponent from winning, and gravitate toward the center of the game board, as a human player would.

Using Python, we implemented a version of the Hint Factory for Connect Four. Our implementation follows a greedy policy that prioritizes the highest immediate reward, regardless of what the future rewards might be. While this may seem like an impatient policy, the adversarial and stochastic nature of Connect Four makes policies relying on potential future rewards more risky. With our aforementioned policy, the value for the states is the reward of the greatest state-action pairing. It indicates which action to select, based on the calculated rewards with transition probabilities, that leads to a state most commonly found on the path to a winning game.

3 Results and Discussion

Our dataset consists of 5,000 Connect Four games, with an average of twenty-six turns (13 states) per game. If we use that average and process all 5,000 games, it results to 65,000 states. However, when we process all 65,000 states for the games, only 2,606 (4.15%) of that total are unique game states. Table 1 summarizes the unique states encountered at different processing increments of the total games.

Table 1. Number of unique states generated from games played in a simulated study.

Games processed	Unique states	Percent increase
100	797	–
500	1,531	92.10
1,000	1,865	21.82
2,000	2,197	17.80
3,000	2,386	8.60
4,000	2,510	5.20
5,000	2,606	3.82

As noted, the percentage of new states encountered continues to decrease, as expected, as we introduce more games. While there are over 4.5 trillion legal board states, many of them will not be reached if players are following general strategies of winning. For instance, human players might follow the strategy of trying to maximize their game pieces in the center columns, causing many of the states to revolve around a similar configuration. This is the case for our agents, both of which follow the same Minimax algorithm and heuristic function in attempts to win. The observed games begin randomly, but the agent's moves tend to gravitate toward the center as they follow the heuristic for winning, much like a human player might.

Using the first 1,000 games to build the model, we were able to provide hints for roughly 50,000 of the 52,000 states that were present in the following 4,000 games. The 52,000 states making up the later 4,000 games includes repeated states, as some of the games followed the same sequence. For our 5,000 observed games, 1,123 of them were unique. This reduces the space from 65,000 total states to 14,600 states if we only include unique games. Building the model based off the first 1000 random games recorded yields 1,865 unique states for our system. Using that data, we were able to provide a hint, suggesting the next optimal move to the user, for all but 1,931 of the encountered states in the remaining 4,000 games. That results in this implementation being able to provide a hint 87.23% of the time when using just the first 20% of the recorded data.

Next-step hints, such as the ones generated by our system, are often not as descriptive as instructor generated ones, causing some educators to doubt their effectiveness for learning. However, such hints have been proven effective, as in a study by Rivers (2017) which resulted in students spending 13.7% less time on practice while achieving the same learning results. Paquette et al. (2012) found that next-step hints yielded increased learning gains within an ITS, compared to offering only flag feedback, and that they were as efficient and appreciated as instructor generated hints. In stochastic learning environments that often lack hints altogether, generating next-step hints using Hint Factory techniques can improve educational effectiveness and easily integrates into an ITS or other educational system.

Increasing complexity is brought on from adversarial agents when a policy is being determined for the system. An adversarial agent creates uncertainty on what the state will be, even when the system suggests a particular action for the player. In the case of our Connect Four implementation, it is not truly random as a heuristic is being followed via the Minimax algorithm. We can offer a hint, suggesting where the player should place their game piece, based on what we have previously seen and its success. However, we have no other source that suggests what action the adversarial agent will take, in this case the column into which they will drop their game piece. This type of uncertainty can make certain policies seemingly optimal, but they might not be as successful when different data is utilized to build the model or when the adversarial agent plays more randomly. Such uncertainty has previously made these domains difficult to provide next-step hints for, as the following state remains unknown.

Hints are effective for student learning and have been proven effective in open-ended domains (Stamper et al. 2011; Rivers 2017), yet their data-driven implementation remains difficult in complex stochastic environments such as immersive training simulation environments One instance of this in chemistry is the VLab, a virtual

environment for students to conduct experiments and interpret the reactions (Tsovaltzi et al. 2010). If students are working on a stoichiometry problem, the reaction will vary based on how much and what type of a certain chemical they add, thus dictating how much of another substance they need to add next. Detailed hints for a system may be hard to generate because they are dependent on the many combinations of chemical type and quantity that could be added. However, it would be a good fit for our techniques, using either collected or simulated student data, to provide hints.

In physical stochastic environments, such as driving a car or operating a robotic arm, many adversarial forces are at play, such as objects in the path or even gravity. Next-step hints in such domains can provide needed decision support and be generated using similar methods as applying the Hint Factory to Connect Four, without requiring mass amounts of data or computational power. In the realm of robotics, Lego Mindstorms have been a popular and effective mechanism for teaching middle and high school students programming and computational thinking (Mosley and Kline 2006). Such hints could assist students in the debugging of their robots, as they get them to perform a given task, taking into account the physical forces they encounter. Similar techniques could be applied to programmatic physics engines used for student experimentation, which often have many reactive forces at play.

4 Conclusion and Future Work

The stochastic feature of adversarial game environments naturally fit the Hint Factory's underlying MDP mechanisms. The main contribution of this work is to show how the Hint Factory can be used to create a hint and feedback policy in an adversarial game, that can be extended to more complex educational domains. The novel implementation in this stochastic domain shows that with a relatively small selection of the overall state space, we can provide coverage to provide hints and feedback for a large percentage of commonly seen states. This has real implications for many other multiagent and adversarial game and learning environments.

We see some obvious next steps for this work in other multi-agent environments, particularly simulations of the physical world where many forces are at play. Determining what to track as part of the state and what encompass an action will need to be addressed as we move into more complex stochastic domains. Next we plan to execute experiments using the hint generations to see if we get similar gains to the Hint Factory implementations in deterministic domains. We also want to look at how human users learn from the suggested hints and mimic any strategies being conveyed.

References

Aleven, V., McLaren, B., Roll, I., Koedinger, K.: Toward tutoring help seeking. In: Lester, J.C., Vicari, R.M., Paraguaçu, F. (eds.) ITS 2004. LNCS, vol. 3220, pp. 227–239. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30139-4_22

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

Allis, L.V.: A knowledge-based approach of connect-four. Vrije Universiteit, Subfaculteit Wiskunde en Informatica (1988)

Anggara, K.: Connect Four (2018). https://doi.org/10.5281/zenodo.1254572

Barnes, T., Stamper, J., Croy, M.: Using Markov decision processes for automatic hint generation. In: Handbook of Educational Data Mining, 467 (2011)

Edelkamp, S., Kissmann, P.: Symbolic classification of general two-player games. In: Dengel, A. R., Berns, K., Breuel, T.M., Bomarius, F., Roth-Berghofer, T.R. (eds.) KI 2008. LNCS (LNAI), vol. 5243, pp. 185–192. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85845-4_23

Edwards, W.: The theory of decision making. Psychol. Bull. **51**(4), 380 (1954)

Koedinger, K.R., Stamper, J.C., McLaughlin, E.A., Nixon, T.: Using data-driven discovery of better student models to improve student learning. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 421–430. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_43

Mosley, P., Kline, R.: Engaging students: a framework using lego robotics to teach problem solving. Inf. Technol. Learn. Perform. J. **24**(1), 39–45 (2006)

Paquette, L., Lebeau, J.-F., Beaulieu, G., Mayers, A.: Automating next-step hints generation using ASTUS. In: Cerri, S.A., Clancey, W.J., Papadourakis, G., Panourgia, K. (eds.) ITS 2012. LNCS, vol. 7315, pp. 201–211. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30950-2_26

Razzaq, L., Heffernan, N.T.: Hints: is it better to give or wait to be asked? In: Aleven, V., Kay, J., Mostow, J. (eds.) ITS 2010. LNCS, vol. 6094, pp. 349–358. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13388-6_39

Rivers, K.: Automated Data-Driven Hint Generation for Learning Programming (2017)

Shih, B., Koedinger, K.R., Scheines, R.: A response time model for bottom-out hints as worked examples. In: Handbook of Educational Data Mining, pp. 201–212 (2011)

Stamper, J.C., Eagle, M., Barnes, T., Croy, M.: Experimental evaluation of automatic hint generation for a logic tutor. In: Biswas, G., Bull, S., Kay, J., Mitrovic, A. (eds.) AIED 2011. LNCS (LNAI), vol. 6738, pp. 345–352. Springer, Heidelberg (2011a). https://doi.org/10.1007/978-3-642-21869-9_45

Stamper, J., Barnes, T., Croy, M.: Enhancing the automatic generation of hints with expert seeding. Int. J. AI Educ. **21**(1–2), 153–167 (2011b)

Stamper, J., et al.: PSLC DataShop: a data analysis service for the learning science community. In: Aleven, V., Kay, J., Mostow, J. (eds.) ITS 2010. LNCS, vol. 6095, p. 455. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13437-1_112

Stamper, J., Barnes, T., Lehmann, L., Croy, M.: The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In: 9th International Conf on Intelligent Tutoring Systems Young Researchers Track, pp. 71–78 (2008)

Stamper, J.C., Barnes, T., Croy, M.: Extracting student models for intelligent tutoring systems. In: Proceedings of the National Conference on Artificial Intelligence, vol. 22, no. 2, p. 1900. AAAI Press; MIT Press, 1999 (2007)

Stamper, J.: Automating the generation of production rules for intelligent tutoring systems. In: Proceedings of the 9th International Conference on Interactive Computer Aided Learning (ICL 2006). Kassel University Press (2006)

Sutton, R., Barto, A.: Reinforcement Learning. The MIT Press, Cambridge (1998)

Thill, M., Koch, P., Konen, W.: Reinforcement learning with N-tuples on the game connect-4. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7491, pp. 184–194. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32937-1_19

Tsovaltzi, D., et al.: Extending a virtual chemistry lab with a collaboration script to promote conceptual learning. Int. J. Technol. Enhanc. Learn. **2**(1–2), 91–110 (2010)