Tackling the Credit Assignment Problem in Reinforcement Learning-Induced Pedagogical Policies with Neural Networks

Markel Sanz Ausin$^{[0000-0002-4526-9252]}$, Mehak Maniktala, Tiffany Barnes, and Min Chi$^{[0000-0003-1765-7837]}$

North Carolina State University, Raleigh, NC, 27695, USA
{msanzau,mmanikt,tmbarnes,mchi}@ncsu.edu

Abstract. Intelligent Tutoring Systems (ITS) provide a powerful tool for students to learn in an adaptive, personalized, and goal-oriented manner. In recent years, Reinforcement Learning (RL) has shown to be capable of leveraging previous student data to induce effective pedagogical policies for future students. One of the most desirable goals of these policies is to maximize student learning gains while minimizing the training time. However, this metric is often not available until a student has completed the entire tutor. For this reason, the reinforcement signal of the effectiveness of the tutor is *delayed*. Assigning credit for each intermediate action based on a delayed reward is a challenging problem denoted the temporal *Credit Assignment Problem* (CAP). The CAP makes it difficult for most RL algorithms to assign credit to each action. In this work, we develop a general Neural Network-based algorithm that tackles the CAP by *inferring* immediate rewards from delayed rewards. We perform two empirical classroom studies, and the results show that this algorithm, in combination with a Deep RL agent, can improve student learning performance while reducing training time.

Keywords: Pedagogical Agent · Credit Assignment Problem · Deep Reinforcement Learning.

1 Introduction

Recent advances in Machine Learning have enabled the creation of algorithms that allow us to optimize certain desired metrics, for a large and diverse pool of users. Reinforcement Learning (RL), in particular, has shown great promise in the last few years, due to its effectiveness in inducing a policy to maximize a reward function while interacting with a non-stationary environment. In recent years, the combination of RL with deep neural networks has enabled solving very complex tasks. Deep RL (DRL) has achieved notable successes in a variety of complex tasks such as robotic control [2] and the game of Go [32]. Despite DRL's great success, there are still many challenges preventing DRL from being applied more broadly in practice, including applying it to educational systems such as Intelligent Tutoring Systems (ITSs).

ITSs and other educational software tools have gained popularity in recent years. These systems allow educators to provide a personalized learning process to each student, without needing to personally supervise the process. These e-learning environments often rely on *pedagogical policies* to decide how each problem or each part of the system is going to be displayed for a given user. The sequential decision-making nature of DRL, combined with its ability to learn from a reward function, makes it a perfect fit to induce pedagogical policies for ITSs and optimize the learning process for each student individually. However, most ITSs have a *delayed reward function* by design. These systems need to assess the overall learning process of each student, and they generally follow a standard structure of pre-test, training on the ITS, and post-test, where the learning improvement is measured. In this situation, discovering which of the tutor's actions are responsible for the delayed outcome can present a challenge. Because of this, the ability of DRL to be broadly effective in real-world applications is still unproven. In such delayed reinforcement tasks, a reward r_t obtained at time t, may have been affected by all the actions leading to that time-step: a_0, a_1, ..., a_{t-1}, and a_t. Assigning credit or blame for each of those actions individually is known as the ***(temporal) Credit Assignment Problem (CAP)*** [19]. The CAP is particularly relevant for real-world tasks, where we need to learn effective policies from small, limited training datasets. In prior work, one way to mitigate the impact of the CAP is to use model-based RL [33, 6] or simulations, which allow collecting vast amounts of data. However, in many real-life domains such as healthcare and education, building accurate simulations is especially challenging because disease progression and student learning are rather complex processes.

The most appropriate rewards to use in education are the student learning outcomes, which are typically unobservable until the entire training process or trajectory is complete. This is because, in human learning progressions, it is difficult to assess student knowledge moment by moment, and more importantly, many instructional interventions that boost short-term performance may not be effective over the long term. To address the CAP, we present a general neural network-based algorithm to infer the immediate rewards from the delayed reward, and then use those inferred immediate rewards for pedagogical policy induction. In this work, we used an ITS that is designed to teach students how to solve logic proofs. We applied DRL to induce a pedagogical policy on one of the most widely studied types of tutorial decisions: whether to present a problem as a **Problem Solving** (PS) or a **Worked Example** (WE) [35, 16, 23, 26, 21, 24, 12, 40, 13, 31, 27, 29].

In this work, we compared two DRL-based pedagogical policies against an Expert baseline policy and a PS-only policy, in two empirical classroom studies with college students. During the Spring 2019 semester, the DRL pedagogical policy first inferred the immediate rewards from the delayed rewards using a Gaussian Processes approach introduced by Azizsoltani et al. [5], and then a DQN agent [20] used those rewards to induce a pedagogical policy, referred to as InferGP in the future. InferGP is compared against an Expert-crafted policy that alternates between PS and WE. Next, during the Spring 2020 semester, we inferred the immediate rewards using InferNet, the algorithm we present in this paper, and then trained a Dueling-DQN agent [39] to induce a policy; InferNet is compared against a PS-only policy because most conventional ITSs are PS only. *As the two DQN-based policies used different scores as reward functions for training, they cannot be directly compared with one another.* Rather, we compare each of those RL-induced policies to the two control groups: Expert and PS-only. Our results show that while no significant difference was found between InferGP, Expert and PS-only in terms of learning gains or learning efficiency, InferNet outperforms the Expert group in terms of learning performance, and InferNet also outperforms the PS-only group in terms of learning efficiency. In short, our proposed InferNet in conjunction with a Dueling-DQN policy results in better and more efficient learning than traditional pedagogical strategies such as Expert-crafted policies or PS-only policies.

2 Background and Related Work

Prior research has applied both online RL and offline RL to induce data-driven pedagogical policies. In *online RL*, the agent learns a policy while interacting with either real or simulated student data, while offline RL approaches "use previously collected samples, and generally provide robust convergence guarantees" [25] and thus, the success of these offline RL approaches depends heavily on the quality of the training data. Furthermore, prior work can be divided into traditional RL vs. DRL approaches. In the former, for instance, Iglesias et al. applied Q-learning to induce policies for efficient learning [10, 11]. More recently, Rafferty et al. applied an online partially observable Markov decision process (POMDP) to induce policies for faster learning [22]. Shen et al. employed offline value iteration and least square policy iteration to induce a pedagogical policy that improved student learning [30, 28]. Chi et al. applied offline policy iteration to induce a pedagogical policy aimed at improving students' learning gain [7]. Mandel et al. [15] used an offline POMDP to induce a policy which aims to improve student performance in an educational game. All the models described above were evaluated in classroom studies and were found to yield certain improved student learning or performance relative to a baseline policy.

The DRL approaches have been motivated by the recent growth in using Deep Neural Networks as function approximation. For instance, the Deep Q-Network (DQN) algorithm [20] takes advantage of convolutional neural networks to learn to play Atari games observing the pixels directly. Since then, DRL has achieved success in various complex tasks such as the games of Go [32], Chess/Shogi [33], Starcraft II [37], and robotic control [2]. One major challenge of these methods is *sample inefficiency*, where RL policies need large sample sizes to learn optimal, generalizable policies. DRL has also been applied to ITSs. Wang et al. applied an online DRL approach to induce a policy for adaptive narrative generation in an educational game using simulations [38]; the resulting DRL-induced policies were evaluated via simulations only. Sanz Ausin et al. used offline DRL to induce pedagogical policies and showed that they can improve student learning, and can be more effective than expert-designed baseline policies [3, 4]. Much prior

Algorithm 1 InferNet + DRL Offline

```
1: Input: Training dataset $D$, Number of training steps $K$
2: for $step \leftarrow 1$ to $K$ do
3:     Sample mini-batch of episodes $B \sim D$ with Delayed Rewards $R_{del}$
4:     Train InferNet on $B$: $L(\theta) = (R_{del} - \sum_{t=0}^{T-1} f(s_t, a_t)|\theta))^2$
5: end for
6: for $ep \leftarrow 1$ to $|D|$ do
7:     Use the trained InferNet to infer immediate rewards for episode $ep$
8:     Replace original rewards with the new InferNet rewards
9: end for
10: Train DRL agent
```

work has induced a pedagogical policy by using DRL directly, while this work combines a mechanism that tackles the CAP, and a DRL algorithm to induce more effective policies.

3 InferNet

The ultimate goal of InferNet is to tackle the temporal CAP by inferring the immediate rewards from the delayed rewards. We model the environment as a standard Markov Decision Process. At time-step t, the environment is in some state s_t, the agent takes an action a_t, and receives a scalar reward r_t, which in the case of delayed rewards is zero unless it is the last reward in the episode, i.e., at the end of the entire trajectory (the delayed reward). We denote the immediate rewards as r and the delayed rewards as R_{del}.

The idea behind InferNet is rather straightforward. It uses a deep neural network to predict the immediate reward at each time-step, for an episode that contains T steps. At each time step t, InferNet receives a state s_t and its corresponding action a_t as inputs, and it outputs the predicted scalar reward r_t for that time-step: $r_t = f(s_t, a_t|\theta)$, where θ indicates the neural network parameters (weights and biases). To train the neural network, InferNet distributes the final delayed reward among all the states in the episode. More precisely, the neural network is trained to predict the immediate rewards from the delayed reward with a constraint: *the sum of all the predicted immediate rewards in each episode must be equal to the delayed reward of that episode:* $R_{del} = f(s_0, a_0|\theta) + f(s_1, a_1|\theta) + ... + f(s_{T-1}, a_{T-1}|\theta)$. By doing so, the network is tasked with modeling the reward function, conditioned on the state and actions that were passed as inputs. InferNet is trained by minimizing the loss between the sum of predicted rewards and the delayed reward.

For the implementation, the TimeDistributed layer available on TensorFlow Keras [1, 8] was employed. This layer allows repeating the same neural network operation across multiple time-steps, sharing weights across time, and we use it to pass the entire episode at once to the neural network, as a sequence of states and action pairs. It should be noted that there is no internal state in InferNet, despite sharing weights across time as in a recurrent neural network.

Each predicted reward is only dependent on the state and action passed as inputs at that timestep. The loss function that is used to train InferNet is shown in Equation 1. Algorithm 1 shows the pseudo-code for training InferNet offline in conjunction with a DRL algorithm.

$$Loss(\theta) = (R_{del} - \sum_{t=0}^{T-1} f(s_t, a_t|\theta))^2 \tag{1}$$

4 GridWorld With Delayed Reinforcement

The effectiveness of InferNet is investigated on a simple GridWorld task where the immediate rewards are known. This allows us to compare the predicted inferred rewards to the true immediate rewards, and measure the error. This environment consists of a 14x7 grid, with five positive rewards (+1) and four negative rewards (-1), located randomly, but always in the same locations. All other states have a reward of zero. The initial state is located at the bottom-right corner of the grid, and the agent's goal is to reach the terminal state, located at the top-left corner while collecting the positive rewards and avoiding the negative ones. The three available actions are to move up, left, and down. The highest total return that can be collected is +5, while the lowest one is -4.

We compare four reward settings: 1) *Immediate rewards:* when available, they are the gold standard. 2) *Delayed rewards:* these rewards are used as a baseline; here all the intermediate rewards will be zero and the delayed reward that indicates how good or bad the intermediate actions are is provided at the end of the episode. In other words, we simulate the delayed rewards by "hiding" the immediate rewards and providing the sum of all the immediate rewards at the end of the episode. 3) *InferGP rewards:* the inferred immediate rewards using the GP algorithm proposed in [5]. 4) *InferNet rewards:* the inferred immediate rewards through InferNet.

In this experiment, we compared different reward settings using both online and offline RL. InferNet can be trained online and offline, while InferGP can only be applied for offline RL. For online RL, we used an online RL algorithm known for being capable of solving the CAP, the TD(λ) algorithm; while for offline RL, we used Q-learning, which is one of the best known RL approaches.

Online TD(λ): TD(λ) is known to be one of the strongest methods to solve the CAP [34] in that it takes advantage of the benefits of Temporal Difference (TD) learning methods, and includes eligibility traces, which allows the agent to look at all the future rewards to estimate the value of each state. Here we compared InferNet against the delayed and immediate rewards because InferGP cannot be applied for online RL. Figure 1 (Left) shows that by minimizing the training error in Eq. 1 (the difference between the delayed reward and the sum of inferred immediate rewards) (red line), InferNet minimizes the true error (the difference between the inferred immediate rewards and the true immediate rewards) (blue line) when trained online. This shows that our method can effectively approximate the true immediate rewards from the delayed reward.

Figure 1 (Right) shows that when the rewards are delayed, TD(λ) is not able to learn as effectively as the agent with the true immediate rewards. However, by

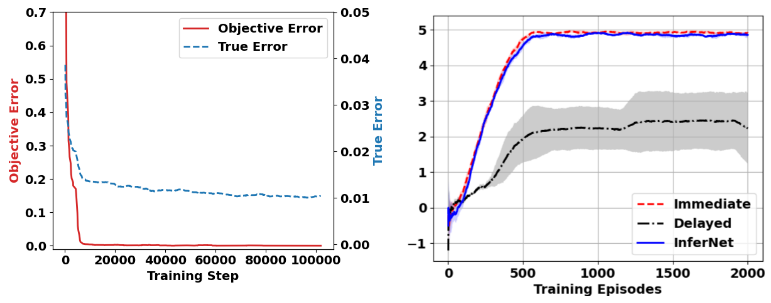

Fig. 1. Online Training. Left: Training InferNet online. Right: Performance of a TD(λ) agent on the GridWorld environment.

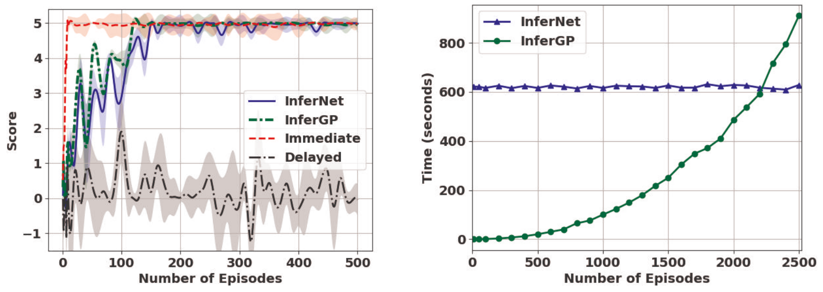

Fig. 2. Offline Training. Left: Performance of Q-Learning agents as a function of the number of training episodes. Right: Empirical time complexity comparison between InferNet and InferGP.

applying InferNet first on the delayed rewards, the InferNet agent achieves the same performance as using the immediate rewards; both converge to the optimal policy. Each experiment is repeated five times with different random seeds, and Figure 1 (Right) shows the mean and standard deviation of those runs.

Offline Q-Learning: In this experiment, we compared all four reward settings. We first generate random gameplay data with immediate rewards from the GridWorld; then sum the immediate rewards in each episode to get its corresponding delayed reward; and finally, apply InferGP and InferNet to infer the corresponding rewards from the delayed reward. To compare the four reward settings, we train a tabular Q-learning agent offline for 5000 iterations on the dataset corresponding to each reward setting. For each of the four reward settings, once its corresponding RL policy is induced, its effectiveness is evaluated by interacting with the GridWorld environment directly for 50 episodes. Figure 2 (left) shows the mean and standard deviation of the performance of the agent, as a function of the number of episodes available in the training dataset. It shows that, as expected, the delayed policy performs poorly, while the Immediate policy can converge to the optimal policy after only 10 episodes of data; additionally, InferNet and InferGP are comparable and both can converge to the optimal policy but they need more training data (around 150 episodes) than the Immediate policy.

Time Complexity: Despite the fact that the performance of InferNet and InferGP is comparable for offline RL, Figure 2 (Right) shows that the training time of InferGP increases cubically $O(n^3)$ as the training data increases, while InferNet has a time complexity of $O(n)$, where n is the amount of training data. This is because InferNet only needs to be trained for a constant number of epochs. Furthermore, InferGP has an asymptotic space complexity of $O(n^2)$, while InferNet has a space complexity of $O(f * l)$, where f is the number of features in the state and action that are passed as inputs, and l is the length of the episode that is passed as input.

In short, InferNet is equivalent to InferGP in performance (as shown by the results in the GridWorld), but its time and space complexity are much better than those of InferGP. InferNet can be applied for both offline and online RL, which makes it much more effective and general, and RL algorithms can benefit greatly from using it to tackle the CAP.

5 Pedagogical Policy Induction

Next we describe how we use InferNet to induce pedagogical policies for an intelligent tutor. We focus on training a pedagogical policy to decide how to show a problem in one of two ways: Problem-Solving (PS) vs. Worked Example (WE). If PS is chosen, students are shown a problem, which they need to complete. In WE, the students are provided with an expert, step-by-step solution to the problem. A great deal of prior research has investigated the effectiveness of PS vs. WE as educational interventions [35, 18, 17, 16, 23, 26, 21, 24]. In general, evidence indicates that showing WEs can significantly reduce the total time on task while not hurting the learning performance too much [18, 17, 16]. On the other hand, alternating between PS and WE can be more effective than PS only [35, 16, 23, 26, 21, 24]. Despite all the prior studies, there is no clear consensus about how or when these two interventions should be combined to optimize student learning. As a result, most existing ITSs always choose PS [14, 36].

Training Corpus: Our training dataset contains 786 student trajectories, collected over five different semesters. Students spend around 2-3 hours on the ITS completing problems. To represent the state of the learning environment, 142 features from five categories are extracted. We have 10 Autonomy features describing the amount of work done by the student; 29 Temporal features including total time spent, time spent on PS, time spent on WE, and so on; 35 Problem-Solving features describing the difficulty of the problem, the number of easy and difficult problems solved, and so on; 57 Performance features such as the number of incorrect steps; and 11 Hint-related features including the total number of hints requested, among others.

Reward Functions: Our goal is to create an RL-induced pedagogical policy to improve student Learning Gain while minimizing the training time. In other words, we want to maximize learning efficiency. In our empirical studies with students, we used two different reward functions and both rewards are only calculated for the post-test problems. The first one, which we will denote as S19 (due to the semester and year when it was used, Spring 2019), uses the number of incorrect rule applications made by the students, as well as the speed, measured by the time spent on the post-test problems. The second reward function, denoted S20, uses the solution length (number of logic statements in the solution), the solution accuracy (proportion of correct rule applications), and the speed. In this work, we applied InferGP to infer the immediate rewards from the delayed rewards using the S19 scoring metric; and InferNet to infer the immediate rewards from the delayed rewards using the S20 scoring metric. Once the inferred immediate rewards are inferred, we train a DQN-based agent [20] to induce the corresponding policies.

Deep Q-Network (DQN) [20] is a version of Q-Learning which uses neural networks for function approximation. DQN uses two neural networks with identical architectures. The main network (represented by the weights θ) is used to estimate the Q-values of the current state s; while the target network (represented by the weights θ^{-}) is used to estimate the Q-values of the next state s' in the Bellman Equation: $Q(s, a|\theta) = r + \gamma \max_{a'} Q(s', a'|\theta^{-})$

Dueling-DQN [39] is an improved version of DQN that splits the Q-value estimation into a value function and an advantage function, and then sums both of them to get the final Q-values. The relation between the value $V(s)$, advantage $A(s, a)$ and Q-value $Q(s, a)$ is defined as A(s, a) = Q(s, a) - V(s). We combined the Dueling-DQN algorithm with a Long Short-Term Memory (LSTM) [9] neural network, which is suitable for tasks with long temporal dependencies.

6 Empirical Experiments

Our ITS teaches how to solve logic proofs. It is used as a graded homework assignment in the undergraduate Discrete Mathematics class at NC State University. To complete a problem, students must iteratively apply rules to logic statement nodes in order to derive the conclusion node. The system automatically checks the correctness of each step and provides immediate feedback on any rule that is applied incorrectly. The ITS consists of a pre-test section, a training phase, and a post-test. The pre-test is used to evaluate the incoming knowledge of the students, and it contains four problems. The pedagogical policy does not take any decisions here. The training phase contains five levels, with four problems per level. The pedagogical policy decides whether to show PS or WE to each student during this phase. However, the policy must follow some constraints determined by the course instructor, who is a professor with over 15 years of experience in the field, to guarantee that every student sees at least one PS and one WE per level. Finally, the post-test consists of six problems, which are used to evaluate the improvement in each student's performance, after undergoing the training stage. The post-test is designed to evaluate the skills of each student, following the requirements of the course. In the end, a score is assigned to each student using one of the two reward functions described in section 5.

We performed two empirical experiments with college students, one in Spring of 2019 (S19) and the other one in Spring of 2020 (S20). For the S19 study, our pedagogical policy was determined by a DQN agent, and the immediate rewards in the training dataset were inferred using InferGP. In the S20 study, we used a Dueling-DQN agent, with immediate rewards inferred using the InferNet algorithm. Both agents used the exact same neural network architecture and hyper-parameters. *It is important to note that the reward functions of S19 and S20 are not directly comparable to each other and thus the induced InferGP and InferNet policies cannot be directly compared to each other, because they were trained to maximize different metrics.*

In the S19 study, we denote our groups as *InferGP*, and *Expert* (for the expert-designed policy that alternates between PS and WE); 64 students were randomly assigned to the two groups and 53 students completed the tutor, with $N = 30$ for InferGP and $N = 23$ for Expert. For the S20 study, we denote our groups as *InferNet*, and *PS-only* (for the policy that always provides PS). 84 students were randomly assigned to the two groups and 74 students completed the tutor, with $N = 36$ for InferNet and $N = 38$ for PS-only. A χ^2 test showed no significant difference between the completion rates of the four different groups: $\chi^2(3, N = 148) = 0.598, p = 0.896$.

7 Results

We analyzed two key metrics that allow us to evaluate the performance of the students and measure their learning: *post-test performance* and *learning efficiency*. Post-test performance evaluates how much the students have learned after using the ITS during the training phase. The learning efficiency also accounts for the time spent in the training phase; it divides the post-test score by the training time, which results in a measurement of how much time they needed to reach a certain knowledge level. We want our policies to help the students learn as much as possible in as little training time as needed.

InferGP: In this analysis, we compared the performance of the *InferGP* policy against the *Expert* and the *PS-only* policies using the S19 scoring metric. A one-way ANOVA test showed no significant difference in the pre-test scores among the three groups: $F(2, 88) = 0.202, p = 0.650$. That is, our pre-test analysis shows that all three groups were balanced in incoming competence.

Next, we analyzed the post-test score performance. A one-way ANCOVA test using the group as a factor and the pre-test score as a covariate showed no significant difference in the post-test scores: $F(2, 87) = 0.019, p = 0.889$. When analyzing the learning efficiency, a one-way ANCOVA test using the group as a factor and the pre-test score as a covariate, also showed no significant difference in the post-test learning efficiency: $F(2, 87) = 2.017, p = 0.159$. In short, our analysis found no significant differences between the students in the InferGP group and the PS-only and Expert groups.

InferNet: Table 1 shows the mean and SD of the performance of the *InferNet* policy against the *Expert* and the *PS-only* policies using the S20 scoring metric. A one-way ANOVA test showed no significant difference in the pre-test scores among the three groups: $F(2, 93) = 1.099, p = 0.297$. Again, our pre-test analysis shows that all three groups were balanced in incoming competence.

We analyzed the post-test score performance. A one-way ANCOVA test using the group as a factor and the pre-test score as a covariate showed a marginal

Table 1. Results by group for the InferNet study.

	Pre-Test Score	Post-Test Score	Learning Efficiency
InferNet	**0.73** (0.27)	**0.72** (0.09)	**0.34** (0.27)
PS-only	0.67 (0.25)	0.70 (0.12)	0.18 (0.16)
Expert	0.67 (0.27)	0.51 (0.10)	0.23 (0.17)

difference in the post-test scores: $F(2,92) = 3.182, p = 0.077$. Subsequent pairwise one-way ANCOVA tests showed a significant difference between the PS-only and the Expert groups ($F(1,57) = 42.336, p < 0.001, d = 1.720$) as well as between the InferNet group and the Expert group ($F(1,55) = 58.200, p < 0.001, d = 2.207$); no significant difference was found between InferNet and PS-only ($F(1,69) = 0.331, p = 0.567$). Finally, we analyze the learning efficiency. A one-way ANCOVA test using the group as a factor and the pre-test score as a covariate showed a significant difference in the post-test learning efficiency: $F(1,92) = 8.839, p = 0.003$. Subsequent pairwise one-way ANCOVA tests showed a significant difference in the learning efficiency between the InferNet and PS-only groups ($F(1,69) = 7.910, p = 0.006, d = 0.721$) and no significant difference was found between InferNet and Expert ($F(1,55) = 2.340, p = 0.132$) or between PS-only and Expert ($F(1,57) = 1.489, p = 0.227$).

To summarize, our results show that the students in the InferNet group achieved a significantly superior post-test score performance than the students in the Expert group, and they were also significantly more efficient than the students in the PS-Only group. This means that they learned more than the students in the Expert group, and they learned more in less time than the students in the PS-only group.

8 Conclusion

In this work, we developed a new method, InferNet, to solve the temporal CAP and help RL agents learn more effectively in delayed reinforcement tasks. We compared our method to immediate and delayed rewards, as well a previous method denoted InferGP, in a simulated GridWorld task, both online and offline, and showed that InferNet can effectively infer the true immediate rewards from the delayed rewards. We also showed that the InferNet rewards can be more effective than the delayed rewards in all cases. Furthermore, we evaluated the effectiveness of the InferNet rewards in two empirical classroom studies with real students, and the results showed that when combining a Deep RL agent with InferNet, the students in the InferNet group achieved a significantly superior post-test score performance than the students in the Expert group, and they were also significantly more efficient than the students in the PS-Only group. These empirical results indicate that our method is effective at helping students learn more in less time. Our method provides a robust and general way to induce a pedagogical policy that can improve student learning.

Acknowledgements: This research was supported by the NSF Grants: #1726550, #1651909, #1937037 and #2013502.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Andrychowicz, M., Baker, B., et al.: Learning dexterous in-hand manipulation. arXiv:1808.00177 (2018)
3. Ausin, M.S.: Leveraging deep reinforcement learning for pedagogical policy induction in an intelligent tutoring system. In: In: Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019), (2019)
4. Ausin, M.S., Maniktala, M., Barnes, T., Chi, M.: Exploring the impact of simple explanations and agency on batch deep reinforcement learning induced pedagogical policies. In: International Conference on Artificial Intelligence in Education. pp. 472–485. Springer (2020)
5. Azizsoltani, H., et al.: Unobserved is not equal to non-existent: Using gaussian processes to infer immediate rewards across contexts. In: In Proceedings of the 28th IJCAI (2019)
6. Chen, B., Xu, M., Li, L., Zhao, D.: Delay-aware model-based reinforcement learning for continuous control. arXiv preprint arXiv:2005.05440 (2020)
7. Chi, M., VanLehn, K., Litman, D., Jordan, P.: Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. User Modeling and User-Adapted Interaction **21**(1-2), 137–180 (2011)
8. Chollet, F.: Keras. https://keras.io (2015)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Iglesias, A., Martínez, P., Aler, R., Fernández, F.: Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. Applied Intelligence **31**(1), 89–106 (2009)
11. Iglesias, A., Martínez, P., Aler, R., Fernández, F.: Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. Knowledge-Based Systems **22**(4), 266–270 (2009)
12. Ju, S., Chi, M., Zhou, G.: Pick the moment: Identifying critical pedagogical decisions using long-short term rewards. In: Rafferty, A.N., Whitehill, J., Romero, C., Cavalli-Sforza, V. (eds.) Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020. International Educational Data Mining Society (2020), https://educationaldatamining.org/files/conferences/EDM2020/papers/paper_167.pdf
13. Ju, S., Zhou, G., Azizsoltani, H., Barnes, T., Chi, M.: Identifying critical pedagogical decisions through adversarial deep reinforcement learning. In: EDM. International Educational Data Mining Society (IEDMS) (2019)
14. Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education (IJAIED) **8**, 30–43 (1997)
15. Mandel, T., Liu, Y.E., Levine, S., Brunskill, E., Popovic, Z.: Offline policy evaluation across representations with applications to educational games. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. pp. 1077–1084. International Foundation for Autonomous Agents and Multiagent Systems (2014)
16. McLaren, B.M., van Gog, T., et al.: Exploring the assistance dilemma: Comparing instructional support in examples and problems. In: Intelligent Tutoring Systems. pp. 354–361. Springer (2014)

17. McLaren, B.M., Isotani, S.: When is it best to learn with all worked examples? In: AIED. pp. 222–229. Springer (2011)
18. McLaren, B.M., Lim, S.J., Koedinger, K.R.: When and how often should worked examples be given to students? new results and a summary of the current state of research. In: Proceedings of the 30th annual conference of the cognitive science society. pp. 2176–2181 (2008)
19. Minsky, M.: Steps toward artificial intelligence. Proceedings of the IRE **49**, 8–30 (1961)
20. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
21. Najar, A.S., Mitrovic, A.: Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum. UMUAI **26**(5), 459–491 (2016)
22. Rafferty, A.N., Brunskill, E., et al.: Faster teaching via pomdp planning. Cognitive science **40**(6), 1290–1332 (2016)
23. Renkl, A., Atkinson, R.K., et al.: From example study to problem solving: Smooth transitions help learning. The Journal of Experimental Education **70**(4), 293–315 (2002)
24. Salden, R.J., Aleven, V., Schwonke, R., Renkl, A.: The expertise reversal effect and worked examples in tutored problem solving. Instructional Science **38**(3), 289–307 (2010)
25. Schwab, D., Ray, S.: Offline reinforcement learning with task hierarchies. Machine Learning **106**(9-10), 1569–1598 (2017)
26. Schwonke, R., Renkl, A., Krieg, C., Wittwer, J., Aleven, V., Salden, R.: The worked-example effect: Not an artefact of lousy control conditions. Computers in Human Behavior **25**(2), 258–266 (2009)
27. Shen, S., Ausin, M.S., Mostafavi, B., Chi, M.: Improving learning & reducing time: A constrained action-based reinforcement learning approach. In: UMAP. pp. 43–51. ACM (2018)
28. Shen, S., Chi, M.: Aim low: Correlation-based feature selection for model-based reinforcement learning. International Educational Data Mining Society (2016)
29. Shen, S., Chi, M.: Reinforcement learning: the sooner the better, or the later the better? In: UMAP. pp. 37–44. ACM (2016)
30. Shen, S., Chi, M.: Reinforcement learning: the sooner the better, or the later the better? In: UMAP. pp. 37–44. ACM (2016)
31. Shen, S., Mostafavi, B., Lynch, C.F., Barnes, T., Chi, M.: Empirically evaluating the effectiveness of POMDP vs. MDP towards the pedagogical strategies induction. In: AIED (2). Lecture Notes in Computer Science, vol. 10948, pp. 327–331. Springer (2018)
32. Silver, D., Huang, A., Maddison, C.J., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484 (2016)
33. Silver, D., Hubert, T., Schrittwieser, J., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
34. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine learning **3**(1), 9–44 (1988)
35. Sweller, J., Cooper, G.A.: The use of worked examples as a substitute for problem solving in learning algebra. Cognition and Instruction **2**(1), 59–89 (1985)
36. VanLehn, K., Graesser, A.C., et al.: When are tutorial dialogues more effective than reading? Cognitive science **31**(1), 3–62 (2007)

37. Vinyals, O., Babuschkin, I., Czarnecki, W., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**, 350 (2019)
38. Wang, P., Rowe, J., Min, W., Mott, B., Lester, J.: Interactive narrative personalization with deep reinforcement learning. In: IJCAI (2017)
39. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. arXiv:1511.06581 (2015)
40. Zhou, G., Azizsoltani, H., Ausin, M.S., Barnes, T., Chi, M.: Hierarchical reinforcement learning for pedagogical policy induction (extended abstract). In: IJCAI. pp. 4691–4695. ijcai.org (2020)