Reinforcement Learning: the Sooner the Better, or the Later the Better?

Shitian Shen
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
sshen@ncsu.edu

Min Chi
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
mchi@ncsu.edu

ABSTRACT

Reinforcement Learning (RL) is one of the best machine learning approaches for decision making in interactive environments. RL focuses on inducing effective decision making policies with the goal of maximizing the agent's cumulative reward. In this study, we investigated the impact of both immediate and delayed reward functions on RL-induced policies and empirically evaluated the effectiveness of induced policies within an Intelligent Tutoring System called Deep Thought. Moreover, we divided students into *Fast* and *Slow* learners based on their incoming competence as measured by their average response time on the initial tutorial level. Our results show that there was a significant interaction effect between the induced policies and the students' incoming competence. More specifically, Fast learners are less sensitive to learning environments in that they can learn equally well regardless of the pedagogical strategies employed by the tutor, but Slow learners benefit significantly more from effective pedagogical strategies than from ineffective ones. In fact, with effective pedagogical strategies the slow learners learned as much as their faster peers, but with ineffective pedagogical strategies the former learned significantly less than the latter.

Keywords

Reinforcement learning, Pedagogical strategy, Immediate Reward, Delayed Reward, Worked Example, Problem Solving

1 Introduction

Optimal decision making in complex interactive environments is challenging. In Intelligent Tutoring Systems (ITSs), for example, the system's behaviors can be viewed as a sequential decision process where at each step the system chooses an appropriate action from a set of options. *Pedagogical strategies* are policies that are used to decide what action to take next in the face of alternatives. Each of these system decisions will affect the user's subsequent actions and performance. Its impact on outcomes cannot be observed immediately and the effectiveness of each decision is dependent upon the effectiveness of subsequent decisions.

Reinforcement Learning (RL) is one of the best machine learning approaches for decision making in interactive environments. RL focuses on inducing effective decision making policies for an agent with the goal of maximizing the agent's cumulative reward. In many domains RL is applied with immediate reward functions. In an automatic call center system, for example, the agent can receive an immediate reward for every question it asks because the impact of each question can be assessed instantaneously [29]. Immediate rewards are generally more effective than delayed rewards for RL-based policy induction. This is because it is easier to assign appropriate credit or blame when the feedback is tied to a single decision. The more we delay the rewards or punishments, the harder it becomes to assign credit or blame properly.

On the other hand, the most appropriate reward to use in an ITSs is student learning gains which are typically unavailable until the entire training process is complete. This is due to the complex nature of the learning process which makes it difficult to assess students' learning moment by moment and more importantly, many instructional interventions that boost short-term performance may not be effective over the long-term. Therefore, in this study, we explored both immediate and delayed rewards in our policy induction and empirically evaluated the impact of the induced policies on student learning.

Prior research has shown that some learners are less sensitive to the learning environment and can always learn; while others are more sensitive to variations in learning environments and may fail to do so [4]. We refer to the former as high learners and the latter as low learners. It is not fully understood why such differences exist. One hypothesis is that low learners lack crucial skills such as general problem-solving strategies and meta-cognition. In order to be effective and to honor the promises of learning environments, a system should support both high and low learners effectively, especially the low learners. It is our hypothesis that our induced pedagogical strategies may have different impacts on students with different learning competence. More specifically, in this study, we divide students into *Fast* and *Slow* groups based upon their average response time and we found that the RL-induced pedagogical strategies had significantly more impact on Slow learners than on their Fast

UMAP '16, July 13-17, 2016, Halifax, NS, Canada
 ISBN 978-1-4503-4370-1/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2930238.2930247

peers. That is, Slow learners in this study behave more like low learners in that they are more sensitive to effectiveness of the RL-induced pedagogical strategies while Fast learners are more like high learners in that they can learn equally effectively regardless of the pedagogical strategies employed.

We applied RL to induce two sets of policies: Immediate and Delayed. We focused on one important tutorial decision: whether to provide students with a Worked Example (WE) or to require them to engage in a Problem Solving (PS). Our primary research questions are: 1) would our induced policies improve students' learning, especially for Slow learners? 2) which policy, Immediate or Delayed, would be more effective?

2 Related Work

2.1 Applying RL to ITSs

Beck et al [2] investigated application of RL to induce pedagogical policies that would minimize the time students take to complete each problem on AnimalWatch, an ITS that teaches arithmetic to grade school students. In the training phase of their study, they used simulated students. Given that student time can be assessed at each step, they used an immediate reward function. In the test phase, the induced policies were added to AnimalWatch and the new system was empirically compared with the original version of AnimalWatch. They found that the policy group spent significantly less time per problem than their no-policy peers.

Iglesias and her colleagues [6, 7, 8], on the other hand, focused on applying RL to improve the effectiveness of an Intelligent Educational System that teaches students Database Design. They applied Q-learning and used immediate rewards. Their goal when inducing the policy was to provide students with direct navigation support through the system's content with the expectation that this would help students learn more efficiently. Iglesias and her colleagues used simulated students in their training phase much like Beck et al. In the test phase, they evaluated the induced policy empirically with real students. Their results showed that while the policy led to more effective system usage behaviors from students, the students using a system equipped with a policy did not outperform their no-policy peers in terms of learning outcomes.

Martin and Arroyo [10] applied a model-based RL method, Policy Iteration, to induce pedagogical policies that would increase the efficiency of hint sequencing on Wayang Outpost web-based ITS. During the training phase, the authors used a student model to generate the training data for inducing the policies and they used delayed rewards. Here the student model was similar to the simulated students used in Beck et al and Iglesias et al. In the test phase, the induced policies were tested on simulated student model rather than human students.

Tetreault et al [25] used an Intelligent Spoken Tutoring System, ITSPOKE, which teaches students college-level physics [9]. In their work, they used a previously collected set of physics tutorial dialogues as a training corpus and investigated the application of Policy Iteration to induce pedagogical policies from it. The focus of their work was on introducing a novel method for evaluating state representations. Thus they did not measure students' learning gains nor did they compare the policy-equipped system to a prior version. Additionally, note that because the training corpus used in this work was not collected with the goal of exploring the full range of tutorial decisions, the tutor often executed only one type of action for many dialogue states. In this study, they used a delayed reward function.

Finally, Chi et al [16] applied model-based RL to induce pedagogical policies to improve the effectiveness to an Intelligent Natural Language Tutoring System, Cordillera, which teaches students college physics. In that study, they collected an exploratory corpus by training human students on the ITS that makes random decisions and then applied RL to induce pedagogical policies from the corpus they used an exploratory training corpus. Their empirical evaluation showed the induced policies were significantly more effective than the previous ones. In that study, they explored two types of reward functions but both are delayed.

Although there have been other studies on application of RL to ITSs, they mostly involved inducing domain models rather than pedagogical policies. For example, Barnes and Stamper [1, 22] have applied Markov Decision Processes (MDP) to construct problem solutions from existing students' solutions for an ITS called Proofs Tutorial, which teaches college-level discrete mathematics. They used a form of the model-based RL method Value Iteration and the resulting problem solutions were then used to generate hints for new students. They found that the extracted solutions and the proposed hint-generating functions were able to provide hints over 80% of the time for the new students.

In short, RL has been applied to induce pedagogical policies in ITSs but previous research explored either delayed or immediate reward but not both. Moreover, previous research has investigated whether all students' learning performance were improved by the induced RL policies while we hypothesized that certain learners are more likely to be affected by the effectiveness of the induced pedagogical strategies than others. The type of the decision we focused on is: PS vs. WE.

2.2 Problem Solving vs. Worked Example

A great deal of research has investigated the differing impacts of worked examples (WE) and problem solving (PS) on student learning [24, 12, 11, 13, 17, 20, 15, 18]. During PS students are given a training problem which they must solve independently or with partial assistance while during WE students are shown a detailed solution to the problem. McLaren and colleagues compared WE-PS pairs with PS-only [12]. Every student was given a total of 10 training problems. Students in the PS-only condition were required to solve every problem while students in the WE-PS condition were given 5 example-problem pairs. Each pair consists of an initial worked example problem followed by tutored problem solving. They found no significant difference in learning performance between the two conditions, however the WE-PS group spent significantly less time than the PS group.

McLaren and his colleagues found similar results in two subsequent studies [11, 13]. In the former, the authors compared three conditions: WE, PS and WE-PS pairs, in the domain of high school chemistry. All students were given 10 identical problems. As before, the authors found no significant differences among the three groups in terms of learning gains but the WE group spent significantly less time than the other two conditions; and no significant time on task difference was found between the PS and WE-PS conditions.

In a follow-up study, conducted in the domain of high school stoichiometry, McLaren and colleagues compared four conditions: WE, tutored PS, untutored PS, and Erroneous Examples (EE) [13]. Students in the EE condition were given *incorrect* worked examples containing between 1 and 4 errors and were tasked with correcting them. Again the authors found no significant differences among the conditions in terms of learning gains, and as before the WE students spent significantly less time than the other groups. More specifically, for time on task they found that: *WE* < *EE* < *untutored PS* < *tutored PS*. In fact, the WE students took only 30% of the total time that the tutored PS students did.

The advantages of worked examples were also demonstrated in another study in the domain of electrical circuits [28]. The authors of that study compared four conditions: WE, WE-PS pairs, PS-WE pairs (problem-solving followed by an example problem), and PS only. They found that the WE and WE-PS students significantly outperformed the other two groups, and found no significant differences was found among four conditions in terms of time on task.

In short, prior research has shown that problem-level worked examples can be as or more effective than problem solving or alternating problems with examples, and the former can take significantly less time than the latter two [24, 12, 11, 13, 17, 20].

2.3 Fast vs. Slow Learners

We hypothesized that certain learners, specifically low learners, are more sensitive to the effectiveness of pedagogical strategies than high learners. Therefore, we need to first distinguish high and low learners based upon some measurement of incoming competence. One way to measure students' incoming competence is to use their prior performance. In this study while all students received the same initial training on level 1, we cannot use their performance alone as a measure of competence as the material is novel and the data would be noisy. This is because: 1) students may make early errors due to their unfamiliarity with the system; 2) students may refer to external resources for information given that all students get the same training problems on level 1. Thus using their performance alone at Level 1 may obscure the student's true incoming competence.

On the other hand, ever since the mid-1950s, response time has been used as a preferred dependent variable in cognitive psychology [27]. It has primarily been used to assess student learning because response time can indicate how active and accessible student knowledge is. For example, it is shown that response time reveals student proficiency [19] and there was a significant negative correlation between the students' average response time and their final exam scores taken at the end of the semester [5]. With the advent of computerized testing, more and more researchers have begun to use response-time as a learning performance measurement [19]. Therefore in this work we used response time as the measure of initial incoming competence.

More specifically, we used the students' average response time on level 1 to split students into fast and slow groups. We compared the students' average response time on level 1 to the median of all average response times on level 1. Students below the median are classified as fast learners while students above the median are classified as slow learners.

3 Inducing Pedagogical Strategies

Prior RL research has typically used Markov Decision Processes (MDPs) [23] to model user-system interactions. The central idea behind this approach is to transform the problem of inducing effective pedagogical strategies on what action the agent should take at any state into computing an optimal policy in an MDP.

3.1 Markov Decision Process

An MDP is a mathematical framework for representing a RL task. It is defined by a tuple $\langle S, A, T, R\rangle$. $S = \{S_1, S_2, ..., S_n\}$ denotes the state space; $A = \{A_1, A_2, ..., A_m\}$ represents a set of agent's possible actions; $T : S \times A \times S \rightarrow [0, 1]$ is a transition probability table, where each element is $T^a_{S_i S_j} = p(S_j|S_i, a)$ which indicates the probability of transiting from state S_i to state S_j by taking an action a; $R : S \times A \times S \rightarrow \mathbb{R}$ assigns rewards to state transitions given actions. And $\pi : S \rightarrow A$ is defined as a policy, mapping state S into action A with the purpose of maximizing expected reward.

After defining the MDP, we can transfer student-system interaction dialog into the trajectory which can be represented as follows:

$$S_1 \xrightarrow{A_1,R_1} S_2 \xrightarrow{A_2,R_2} S_3 \xrightarrow{A_3,R_3} ... \rightarrow S_N$$

Where $S_i \xrightarrow{A_i,R_i} S_{i+1}$ means that the tutor executed action A_i and received reward R_i under state S_i and then transferred to the next state S_{i+1}. In general, the reward can be divided into immediate and delayed rewards, where immediate reward exists during the state transition process while delayed reward follows terminal state.

3.2 Training Datasets

Our dataset was collected from the Deep Thought (DT) tutoring system [14]. It included a total of 303 undergraduate CS students who used DT as part of class assignment in Fall 2014 and Spring 2015. The average amount of time spent in the tutor was 416.60 minutes. When the students start each new training problem, DT will make a simple decision: should it ask the student to solve the next problem **(PS)**, or should it provide them with a worked example **(WE)**. In order to model the students' learning process, we extracted a total of 134 state feature variables, which can be grouped into the following five categories:

1. **Autonomy (AM):** the amount of work done by the student: such as the number of problems solved so far *PSCount* or the number of hints requested *hintCount*.

2. **Temporal Situation (TS):** the time related information about the work process: such as the average time taken per problem *avgTime*, or the total time for solving a problem *TotalPSTime*.

3. **Problem Solving (PS):** information about the current problem solving context, such as the difficulty of the current problem *probDiff*, or whether the student changes the difficulty level *NewLevel*.

4. **Performance (PM):** information about the student's performance during problem solving: such as the number of right application of rules *RightApp*.

5. **Student Action (SA):** the statistical measurement of student's behavior: such as the number of non-empty-click actions that students take *actionCount*, or the number of clicks for derivation *AppCount*.

3.3 Immediate vs. Delayed Policies

The reward function in DT datasets is calculated based upon the level score $LevelScore_i$ where $i \in [1, 6]$, which is calculated based upon the students' performance on the last problem in each level without receiving any formative feedback from system. As described below, students were trained either on high track or low track on each level (except on level 1) so it is hard to compare their performance directly. Therefore, a student's level score at level i is calculated based on *rank* of the student performance score at level i relative to whole population performance scores at the same level.

Our experimental results indicate that there exists a significant correlation between students' performance on last level ($LevelScore_6$) and their final test score taken at the end of the semester: $R^2 = 0.396$, p-value =1.433e-11. This suggests that the students' level score indeed reflects their knowledge level.

We designed two types of reward: immediate and delayed reward with the goal of measuring the students' learning gains. The immediate reward is defined as $R_i = LevelScore_i - LevelScore_{i-1}$ where $i \in [1, 6], R_1 = LevelScore_1$, it reflects the change in students' performance level by level. The delayed reward is defined as $R_{delay} = LevelScore_6 - LevelScore_1$, which determines the change in students' performance across levels. For convenience, we denote the DT datasets with immediate reward as *DT-Immed* and that with delayed reward as *DT-Delay*. Note that the sum of each student's immediate rewards will equal their final delayed reward. Apart from the reward functions, both two datasets are identical.

Both Immediate and delayed policies were induced using the same general procedure. We apply the ensemble feature selection method to the corresponding dataset. In order to extract the state feature set that can represent learning context compactly and accurately, we set the maximum number of state feature size to be 8. The ensemble method comprises 6 correlation-based methods and 4 RL-based methods and more details are described in [21]. Based upon the extracted feature set, we induce our policy using the toolkit developed by Tetreault and Litman[26]. The effectiveness of policy is valuated by Expected Cumulative Reward (ECR)[3] defined as:

$$ECR = \sum_i \frac{N_i}{N} \times V^{\pi}(S_i)$$

where N denotes the number of initial states in training corpus, N_i means the number of state S_i as initial states. The higher the ECR value of a policy, the better the policy is supposed to perform. Our best Immediate policy has a feature size of 7 and our best Delayed policy has a feature size of 6, as shown in Section 5.

Our primary goal is to empirically evaluate the effectiveness of the RL induced policies. In order to do so, we incorporated them back into the DT tutor and empirically compared them against a baseline policy that makes random decisions. Thus, we have three conditions: Immediate, Delayed and Random. Apart from the differing policies, the remaining components of the system, including the GUI interface, the same training problems, and the tutorial scripts, were the same for all three conditions. Next, we will describe our experiment into details.

4 Experiment

The purpose of this experiment is to compare the student performance on DT using Immediate, Delayed, and Random policies respectively.

4.1 Participants

The study was conducted in "Discrete Mathematics for Computer Science" a course offered at North Carolina State University in the Spring of 2016. 106 undergraduate students were assigned to complete the task as one of their regular homework assignments.

4.2 Conditions

The participants were randomly distributed into three conditions. The group sizes were as follows: $N = 30$ for Random, $N = 38$ for Delayed, and $N = 38$ for Immediate condition[1]. A total of 98 students completed the experiment and were distributed as follows: $N = 28$ for Random, $N = 37$ for Delayed, and $N = 33$ for Immediate. We performed a χ^2 test of independence to examine the relationship between completion rate and condition. We found no significant differences among three groups: $\chi^2(2, N = 106) = 1.40, p = 0.496$.

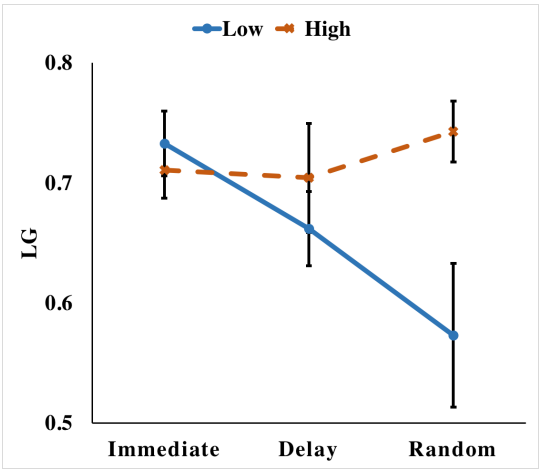
4.3 DT Tutor & Procedure

Deep Thought (DT) is a data-driven ITS that teaches logic proofs and it was used as part of an assignment in undergraduate discrete mathematics course. DT is based upon a data-driven mastery learning system, consisting of 6 strictly ordered levels of proof problems [14]. Each level can be split into two proficiency tracks. The high proficiency track has a smaller number of complex problems, while the low proficiency track contains a larger number of simple problems. The students were required to complete 1-4 problems per level and a total of 7–24 problems overall. All of the students received the same set of problems in level 1. Their initial proficiency is calculated based upon the number of mistakes made on the final problem of level 1. The proficiency reflects how well they understand the knowledge and can apply the logic rules in the proof process. In each sequential level, DT firstly estimate the students' proficiency and then assign them to the high or low track.

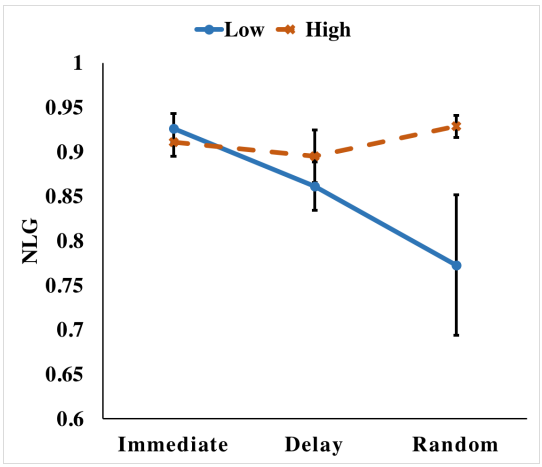
4.4 Performance Measure

When inducing both the Immediate and Delayed policies, we calculated our reward function based upon the students' level scores $LevelScore_i$ where $i \in [1, 6]$. Both Immediate and Delayed policies are induced to maximize the students' improvement from level 1 to level 6. This can be calculated as: $levelScore_6$ - $levelScore_1$. Here we treat $levelScore_6$ as the posttest score, $levelScore_1$ as pretest score, and calculate each student's learning gain as $levelScore_6$ - $levelScore_1$. In addition to raw learning gains, we also used the normalized learning gain (NLG) as the reward value. This measures students' learning gain by considering *their incoming competence* and has been widely used for mearsuring student learning performance in the field of ITS. The NLG is

[1]Note that a slightly smaller portion of students were assigned to the baseline Random condition.



(a) Learning Gain



(b) NLG

Figure 1: Learning Performance across Three Groups

defined as: $NLG = \frac{posttest - pretest}{Max - pretest}$, that is, how much did the student learn given how much he/she can learn. In this study, we calculated NLG as: $NLG = \frac{levelScore_6 - levelScore_1}{MaximumScore - levelScore_1}$. Here $MaximumScore$ is the maximum score a student can get. We will report our students' performance using both the raw learning gain and the NLG. Both scores were normalized to [0,1].

5 Results

We found no significant difference among the three groups in terms of their level 1 score: $F(2, 97) = 0.037, p = 0.964$. As described in Section 2, we subdivided the groups into Fast ($n = 49$) and Slow ($n = 49$) groups based upon their average response time on Level 1. As expected, there was a significant difference between the Fast and Slow students on $LevelScore_1$: $F(1, 98) = 10.05, p = 0.002$. We then partitioned the students into six groups based upon their incoming competence and condition: Immediate-Fast (n = 16), Immediate-Slow (n = 17), Delayed-Fast (n = 17), Delayed-Slow (n = 20), Random-Fast (n = 16), and Random-Slow (n = 12). Again, we find no significant difference among the Slow groups on their level 1 score: $F(2, 46) = 0.56, p = 0.58$. Nor did we find any significant difference among the Fast groups on the same score: $F(2, 46) = 0.64, p = 0.53$.

A two-way ANOVA based upon Condition {Immediate, Delayed, Random} and Incoming Competence {Fast, Slow} showed no significant differences among the three conditions on overall training time: $F(1, 98) = 0.131$, $p = .877$. However there was a significant Incoming Competence effect: the fast learners spent less time on task than the slow learners: $M = 387, SD = 81$ for fast learners and $M = 662, SD = 83$ for the slow learners and the difference was significant: $F(1, 98) = 5.54$, $p = .021$. In addition there was no interaction effect. Therefore, we can conclude that the fast learners spend less time on task than the slow learners across all three conditions.

Next we will report the impact of RL policies on students' performance and then discuss the characteristics of the induced policies.

5.1 Learning Performance

We performed two one-way ANOVAs using condition as the factor and the student's raw learning gain or NLG as the dependent measure respectively. We found no significant difference among the three groups: $F(2, 97) = 1.54$, $p = 0.22$ for raw learning gains and $F(2, 97) = 2.15$, $p = 0.12$ for NLG scores. While no significant difference was found, the comparatively large SD suggests that both fast and slow students may benefit differently from the induced policies. For example, on the raw learning gain scores, $M = 0.72, SD = 0.10$ for the Immediate group, $M = 0.68, SD = 0.16$ for the Delayed group, and $M = 0.67, SD = 0.17$ for the Random group. The same pattern was repeated for NLG.

A two-way ANOVA using Condition {Immediate, Delayed, Random} and Incoming Competence {Fast, Slow} as two factors and the student's raw learning gain or NLG as the dependent measure showed an significant interaction effect, $F(2, 97) = 3.43$, $p = 0.037$ for the raw learning gain and $F(2, 97) = 3.48$, $p = 0.035$ for NLG. Additionally, we also found a significant main effect from the *Incoming Competence*: $F(1, 98) = 4.68$, $p = 0.033$ for the raw learning gain and $F(1, 98) = 4.90$, $p = 0.029$ for the NLG. Therefore the fast learners learned significantly more than the slow learners: $M = 0.719, SD = 0.14$ for the fast learners vs. $M = 0.656, SD = 0.145$ for the slow learners on the raw learning gain. In other words, this results confirmed our assumption that Fast learners can be seen as the high learners while Slow learners can be seen as the low learners. Finally, the main effect of Condition was not significant: $F(1, 98) = 1.54$, $p > 0.05$.

Figure 1 shows that the raw learning gain and NLG results are consistent with our hypothesis: no significant difference was found among the three fast groups on either the raw learning gain or NLG: $F(2, 46) = 0.377$, $p = 0.69$ for the raw learning gain and $F(2, 46) = 0.64$, $p = 0.53$ for NLG respectively.

On the other hand, Figure 1 shows a significant difference among the three slow groups: $F(2, 46) = 3.99$, $p = 0.025$ for the raw learning gain and $F(2, 46) = 3.22$, $p = 0.049$ for NLG. Pairwise t-tests showed that the Immediate-Slow group significantly outperformed the Random-Slow group on

Table 1: Selected features in policies

Feature	Definition	Category
TotalPSTime (f_{I1})	Total time for solving a problem	Temporal Situation
NewLevel (f_{I2})	Whether current solved problem is in the new level	Problem Solving
WrongApp (f_{I3})	Number of wrong application of rules	Performance
TotalWETime (f_{I4})	Total time for working on an example	Temporal Situation
UseCount (f_{I5})	Number of different types of applied rules in Use category	Problem Solving
AppCount (f_{I6})	Number of clicks for derivation	Student Action
NumProbRule (f_{I7})	Number of expected distinct rules for a solved problem	Problem Solving
stepTimeDev (f_{D1})	Step time deviation	Temporal Situation
probDiff (f_{D2})	Difficulty of current solved problem	Problem Solving
symbolicRCount (f_{D3})	Number of whole problems for symbolic representation	Problem Solving
actionCount (f_{D4})	Number of non-empty-click actions that students take	Student Action
SInfoHintCount (f_{D5})	Number of System Information Hint	Student Action
NSClickCountWE (f_{D6})	Number of next step click in Work Example	Student Action

both measures: $t(27) = 2.69$, $p = 0.012$ for the raw learning gain and $t(27) = 2.23$, $p = 0.034$ for NLG. The Immediate-Slow group outperformed the Delayed-Slow group on both measures. However these differences were only marginally significant: $t(35) = 1.67$, $p = 0.098$ for the raw learning gain and $t(35) = 1.94$, $p = 0.060$ for NLG respectively. Furthermore, we found no significant differences between the Delayed-Slow and the Random-Slow groups. Thus, our results suggest that all three Fast groups learned equally well after training on DT while the Slow learners are indeed more sensitive to induced policies. For three slow groups, Immediate policies significantly outperforms Random ones and there is a trend that the Immediate policies beat the Delayed ones while no significant difference between Delayed and Random.

Finally, we compared the fast and slow groups across all three conditions. For the Immediate condition, we found no significant differences among the Immediate-Fast and Immediate-Slow groups on either the raw learning gain or NLG: $F(1, 33) = 0.38$, $p = 0.55$ for the raw learning gain and $F(1, 33) = 0.45$, $p = 0.51$ for NLG. Likewise, for the Delayed condition, no significant difference was found between the Fast and Slow groups: $F(1, 37) = 0.622$, $p = 0.435$ for the raw learning gain and $F(1, 37) = 0.708$, $p = 0.41$ for NLG. Therefore, both fast and slow groups learned equally well when following the RL induced policies.

For Random group, however, the Random-Slow students learned significantly less than their Random-Fast peers: $F(1, 27) = 8.18$, $p = 0.008$ for the raw learning gain and $F(1, 27) = 5.03$, $p = 0.034$ for NLG respectively.

Overall, our results suggest that the Fast learners are not sensitive to the effectiveness of the pedagogical strategy while the Slow learners will learn more with the effective pedagogical strategy. We found that for the Slow learners the Immediate policy is more effective than a Random policy, and is marginally more effective than the Delayed policy.

5.2 Induced Pedagogical Strategy

Immediate Policy. The Immediate policy used seven features f_{I*}, which are listed in Table 1. It's interesting to note that the Immediate policy contained features from every category except Autonomy. Table 2 shows this policy. Each row of table represents one combination of the first four features and each column represents the combination of final three. The black cells indicate that the tutorial action that is associated with the state is PS, while the white

Table 2: Immediate Policy

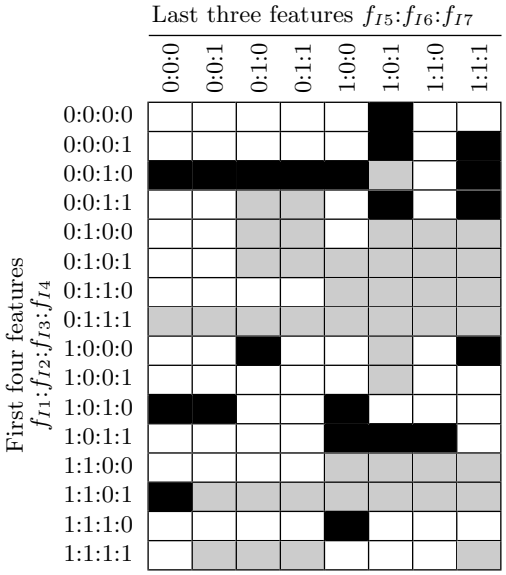


Table 3: Delayed Policy

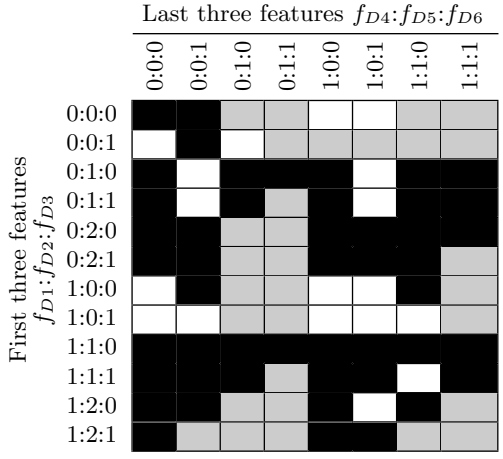


Table 4: Statistical measurement of several features among three groups

Feature	Immediate	Delay	Random	Significance	
TotalCount	19.42(2.07)	20.73(2.08)	19.81(2.09)	$F(2,98) = 3.67,$	$p = 0.03$
PSCount	8.09(1.37)	13.41(1.38)	12.34(1.39)	$F(2,98) = 140.14,$	$p = 0.00$
diffPSCount	4.21(1.90)	7.83(1.91)	6.74(1.92)	$F(2,98) = 32.60,$	$p = 0.00$
WECount	11.33(1.32)	7.32(1.32)	7.47(1.33)	$F(2,98) = 97.11,$	$p = 0.00$
diffWECount	7.78(1.69)	4.38(1.70)	5.50(1.71)	$F(2,98) = 35.78,$	$p = 0.00$

cells indicate the action is WE. The gray cells indicate that no rule was learned for the state. There are a total of 86 rules for the Immediate policy, of which 21 are associated with PS and 65 are associated with WE. More specifically, we found that the states in the row 0:0:1:0 almost always associate with PS; while the states in rows 0:0:0:0, 0:0:0:1, 1:1:1:0 almost always relate to WE; the rules in rows 0:1:0:0, 0:1:0:1, 0:1:1:0, 1:1:0:0, 1:1:1:1 do not contain PS; there are no rules for row 0:1:1:1. In general, the Immediate policy favors WE.

Delayed Policy. The Delayed policy used six features f_{D*}, which are listed in Table 1. They are drawn from the Temporal Situation, Problem Solving and Student Action categories. No features are drawn from the Autonomy and Performance categories. Table 3 shows the Delayed policy, where each row represents one combination of the first three features and each column represents one combination of last three. Each cell has the same meaning as described above. There are a total of 68 rules for the Delayed policy, of which 48 are associated with PS and 20 are associated with WE. More specifically, states in row 1:1:0 only associate with PS; states in rows 0:0:1, 0:1:1, 1:1:1 almost always correspond to PS; while the rules in row 1:0:1 only contain WE. Therefore the Delayed policy is more likely to take PS.

Immediate Policy vs. Delayed Policy. The ECR of the best Immediate policy was 137.97 while the ECR of the best Delayed policy was 14.06. This is likely due to the credit assignment problem. The more we delay success measures from a series of sequential decisions, the more difficult it becomes to identify which of the decision(s) in the sequence are responsible for our final performance. Furthermore, this may explain why, for the slow learners, the low-Immediate students learned more than the low-Random and the low-Delayed students. The former difference was statistically significant while the latter was marginal. We found no significant difference between the low-Random and low-Delayed students.

5.3 Log Analysis

Having compared the individual student's learning performance and the characteristics of the induced policies, this subsection will compare the log file variations across the conditions. More specifically, we focused on the total number of problems that students encountered (TotalCount); the total number of problems that the students solved (PSCount); the total number of WEs reviewed (WSCount); the total number of difficult problems that the students solved (DiffPSCount); and the total number of difficult WEs (DiffWECount).

Table 4 shows comparisons of the different counts across the conditions. A two-way ANOVA using Condition {Immediate, Delayed, Random} and Incoming Competence: {High, Low} on all behavior counts showed a significant main effect for the Condition on all measures. Both the main effect of Incoming Competence and the interaction effect were not significant.

Table 4 summarizes one-way ANOVA comparisons on the different counts among the three conditions. Columns 2-4 list the three groups in comparison and their corresponding mean and SD scores. The last column lists the statistical results of the one-way ANOVA comparisons. Table 4 shows that the Immediate group solved significantly fewer total problems than the Delayed group. The former solved significantly fewer difficult problems than the other two groups. On the other hand, the Immediate group studied significantly more worked examples on both total and difficult problems than the other two groups.

6 Conclusion

In this study, we investigated the impact of different reward functions (Immediate vs. Delayed) on the effectiveness of the induced RL policies. Our results show that the two policies include substantially different state features and that the policies generate different patterns of decisions. The Immediate policies are more likely to give worked examples while the Delayed polices are more likely to require problem-solving. Additionally, the Expected Cumulative Reward (ECR) for our immediate polices was an order of magnitude higher than the delayed ECR.

We also investigated the impact of RL-induced polices on different groups of learners. We divided students into fast and slow learners based upon their average response time in the first level. Our results confirm our hypothesis that the fast learners in all three conditions learned more than their slow peers and that there was no meaningful differences among them across three conditions; the slow learners, on the other hand, were more sensitive to the learning environment. The Random-Slow students learn the least while the Immediate-Slow group learned the most and in fact, it learned as much as their Immediate-Fast peers. Indeed, the Immediate-Slow students learned significantly more than the Random-Slow students, and marginally more than the Delayed-Slow students. Therefore, the Immediate polices appear to be more effective than the Delayed polices and are significantly better than the Random policy.

Finally, our preliminary log analysis showed that students using the Immediate policies studied significantly more worked examples, in terms of both total and difficult problems, than the other two groups. And more importantly, they solved significantly fewer problems, especially difficult problems. Previous research on WE versus PS has primarily relied on fixed or hand-coded adaptive rules to decide whether to present the next question as a WE or PS, this is the first study in which we applied RL to induce adaptive pedagogical strategies directly from students' logs to decide whether

to present the next question as a WE or PS. We showed that the induced policies are indeed effective at improving students' learning especially for Slow learners.

7 Acknowledgements

This research was supported by the NSF Grant 1432156 "Educational Data Mining for Individualized Instruction in STEM Learning Environments".

8 References

[1] T. Barnes and J. C. Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. In *Intelligent Tutoring Systems*, pages 373–382, 2008.

[2] J. Beck, B. P. Woolf, and C. R. Beal. Advisor: A machine learning architecture for intelligent tutor construction. In *AAAI/IAAI*, pages 552–557, 2000.

[3] M. Chi, K. VanLehn, D. Litman, and P. Jordan. 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.

[4] L. J. Cronbach and R. E. Snow. *Aptitudes and instructional methods: A handbook for research on interactions*. New York: Irvington, 1977.

[5] W. J. González-Espada and D. W. Bullock. Innovative applications of classroom response systems: Investigating students' item response times in relation to final course grade, gender, general point average, and high school act scores. *Electronic Journal for the Integration of Technology in Education*, 6:97–108.

[6] A. Iglesias, P. Martínez, R. Aler, and F. Fernández. Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. *Applied Intelligence*, 31:89–106, 2009. 10.1007/s10489-008-0115-1.

[7] A. Iglesias, P. Martínez, R. Aler, and F. Fernández. Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems*, 22(4):266–270, 2009. Artificial Intelligence (AI) in Blended Learning.

[8] A. Iglesias, P. Martínez, and F. Fernández. An experience applying reinforcement learning in a web-based adaptive and intelligent educational system. *Informatics in Education*, 2(2):223–240, 2003.

[9] D. J. Litman and S. Silliman. Itspoke: an intelligent tutoring spoken dialogue system. In *Demonstration Papers at HLT-NAACL 2004*, pages 5–8. Association for Computational Linguistics, 2004.

[10] K. N. Martin and I. Arroyo. Agentx: Using reinforcement learning to improve the effectiveness of intelligent tutoring systems. pages 564–572.

[11] B. M. McLaren and S. Isotani. When is it best to learn with all worked examples? In *Artificial Intelligence in Education*, pages 222–229. Springer, 2011.

[12] B. M. McLaren, S.-J. Lim, and K. R. Koedinger. 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*, pages 2176–2181, 2008.

[13] B. M. McLaren, T. van Gog, C. Ganoe, D. Yaron, and M. Karabinos. Exploring the assistance dilemma: Comparing instructional support in examples and problems. In *Intelligent Tutoring Systems*, pages 354–361. Springer, 2014.

[14] Z. L. Mostafavi Behrooz and T. Barnes. Data-driven proficiency profiling. In *Proc. of the 8th International Conference on Educational Data Mining*, 2015.

[15] A. S. Najar, A. Mitrovic, and B. M. McLaren. Adaptive support versus alternating worked examples and tutored problems: Which leads to better learning? In *User Modeling, Adaptation, and Personalization*, pages 171–182. Springer, 2014.

[16] M. Chi, K. VanLehn, D. J. Litman, and P. W. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Model. User-Adapt. Interact.*, 21(1-2):137–180, 2011.

[17] A. Renkl, R. K. Atkinson, U. H. Maier, and R. Staley. From example study to problem solving: Smooth transitions help learning. *The Journal of Experimental Education*, 70(4):293–315, 2002.

[18] R. J. Salden, V. Aleven, R. Schwonke, and A. Renkl. The expertise reversal effect and worked examples in tutored problem solving. *Instructional Science*, 38(3):289–307, 2010.

[19] D. L. Schnipke and D. J. Scrams. Exploring issues of examinee behavior: Insights gained from response-time analyses. *Computer-based testing: Building the foundation for future assessments*, pages 237–266, 2002.

[20] R. Schwonke, A. Renkl, C. Krieg, J. Wittwer, V. Aleven, and R. Salden. The worked-example effect: Not an artefact of lousy control conditions. *Computers in Human Behavior*, 25(2):258–266, 2009.

[21] S. Shen and M. Chi. Aim low: Correlation-based feature selection for model-based reinforcement learning. 2016.

[22] J. C. Stamper, T. Barnes, and M. J. Croy. Extracting student models for intelligent tutoring systems. pages 1900–1901.

[23] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press Bradford Books, 1998.

[24] J. Sweller and G. A. Cooper. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2(1):59–89, 1985.

[25] J. R. Tetreault, D. Bohus, and D. J. Litman. Estimating the reliability of mdp policies: a confidence interval approach. In *HLT-NAACL*, pages 276–283, 2007.

[26] J. R. Tetreault and D. J. Litman. A reinforcement learning approach to evaluating state representations in spoken dialogue systems. *Speech Communication*, 50(8):683–696, 2008.

[27] R. D. L. V. S. Thomas et al. *Response Times: Their Role in Inferring Elementary Mental Organization: Their Role in Inferring Elementary Mental Organization*. Oxford University Press, USA, 1986.

[28] T. Van Gog, L. Kester, and F. Paas. Effects of worked examples, example-problem, and problem-example

pairs on novices' learning. *Contemporary Educational Psychology*, 36(3):212–218, 2011.

[29] J. D. Williams. The best of both worlds: unifying conventional dialog systems and pomdps. In *INTERSPEECH*, pages 1173–1176, 2008.