# Towards Closing the Loop: Bridging Machine-induced Pedagogical Policies to Learning Theories

Guojing Zhou, Jianxun Wang, Collin F. Lynch, Min Chi
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
{gzhou3,jwang75,cflynch,mchi}@ncsu.edu

## ABSTRACT

In this study, we applied decision trees (DT) to extract a compact set of pedagogical decision-making rules from an original *full* set of 3,702 Reinforcement Learning (RL)-induced rules, referred to as the DT-RL rules and Full-RL rules respectively. We then evaluated the effectiveness of the two rule sets against a baseline Random condition in which the tutor made random yet reasonable decisions. We explored two types of trees (weighted and unweighted) as well as two pruning strategies (pre- and post-pruning). We found that post-pruned weighted trees produced the best results with 529 DT-RL rules. The empirical evaluation was conducted in a classroom study using an existing Intelligent Tutoring System (ITS) named Pyrenees. 153 students were randomly assigned to three conditions. The procedure was the same for all students with domain content and required steps strictly controlled. The only substantive differences between the three conditions were the policy: (Full-RL vs. DT-RL vs. Random). Our result showed that as expected the machine induced policies (Full-RL and DT-RL) are significantly more effective than the random policy; more importantly, no significant difference was found between the Full-RL and DT-RL policies though the number of DT-RL rules is less than 15% of the number of the Full-RL rules and the former group also took significantly less time than the latter.

## 1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) are interactive e-learning environments that support students' learning by providing instruction, scaffolded practice, and on-demand help. The system's behaviors can be viewed as a sequential decision-making process where at each step the system chooses an appropriate action from a set of options. *Pedagogical strategies* are the policies used to decide what action to take next in the face of alternatives. Each system decision will affect the user's subsequent actions and performance. Its impact on outcomes cannot always be immediately observed and the effectiveness of each decision depends upon the effectiveness of subsequent actions. Ideally, an effective learning environment will adapt its decisions to users' specific needs [1, 11]. However, there is no existing well-established theory on how to make these system decisions effectively. Generally speaking, prior research on pedagogical policies can be divided into two general categories: top-down or *theory-driven*, and bottom-up or *data-driven*.

In theory-driven approaches, ITSs employ hand-coded pedagogical rules that seek to implement existing cognitive or learning theories [1, 10, 17]. While existing learning literature gives helpful guidance on the design of pedagogical rules, such guidance is often too general to implement as effective immediate decisions. For example, the aptitude-treatment interaction (ATI) theory states that instructors should match their interventions to the aptitude of the learner [5]. While the principle behind this theory is understandable, it is not clear how to implement that rule for each decision. How do we represent learner's aptitude for each equation, how exact should be the system's adaptation, and so on.

Data-driven approaches, on the other hand, derive pedagogical policies directly from prior data. Here the policies specify the pedagogical decisions at a detailed level. Reinforcement Learning (RL), which we use here, is one popular approach that is able to derive pedagogical policies directly from student-system interaction logs. These policies are defined as a set of state-action mapping rules, which give the best decision to take in each state. The states are typically represented as sets of features and the actions are pedagogical actions such as presenting a worked example (WE) or requiring the student to solve problems (PS). When the system presents a worked example, the students will be given a detailed example showing a complete expert solution for the problem or the best step to take given their current solution state. In Problem Solving, by contrast, students are tasked with solving a problem using the ITS or with completing an individual problem-solving step.

For this project, our original complete RL-induced policy involves the following seven features representing the students' learning process from different perspectives[1].

[1]In the format of: [Feature-Name] (Discretization Procedure): Explanation of the feature.

1. **[nWESincePS]** **(**$0 \rightarrow 0; (0,1] \rightarrow 1; (1,+\infty) \rightarrow 2$**):** The number of worked example (WE) steps received since the last problem solving (PS) step.

2. **[timeInSession]** **(**$[0,2290] \rightarrow 0; (2290,4775] \rightarrow 1; (4775,7939] \rightarrow 2; (7939,+\infty) \rightarrow 3$**):** The total time spent in the current session.

3. **[avgTimeOnStepPS]** **(**$[0,29.01] \rightarrow 0; (29.01, 48.71] \rightarrow 1; (48.71,+\infty) \rightarrow 2$**):** The average amount of time spent on each PS step.

4. **[avgTimeOnStepSessionPS]** **(**$[0,23.51] \rightarrow 0; (23.51,36.56] \rightarrow 1; (36.56,55] \rightarrow 2; (55,+\infty) \rightarrow 3$**):** The average amount of time spent on each PS step in the current session.

5. **[nStepSinceLastWrongKC]** **(**$[0,1] \rightarrow 0; (1,7] \rightarrow 1; (7,25] \rightarrow 2; (25,+\infty) \rightarrow 3$**):** The number of steps received since the last wrong PS step on the current knowledge component (KC).

6. **[nWEStepSinceLastWrong]** **(**$[0,1] \rightarrow 0; (1,4] \rightarrow 1; (4,10] \rightarrow 2; (10,+\infty) \rightarrow 3$**):** The number of WE steps since the last wrong PS step.

7. **[nCorrectPSStepSinceLastWrongKCSession]** **(**$0 \rightarrow 0; (0,3] \rightarrow 1; (3,10] \rightarrow 2; (10,+\infty) \rightarrow 3$**):** The number of correct PS steps since the last wrong PS step on the current KC in the current session.

With this feature set, a state can be represented as a 7-dimensional vector where each element denotes a discretized feature value. Then, the rules can then be represented as:
(0:0:0:0:0:0:0) -> PS
(0:0:0:0:0:0:1) -> PS
(0:0:0:0:0:1:0) -> PS
(0:0:0:0:0:1:1) -> WE
In this study we discretized the features into three-four values producing a seven-feature state. This results in a state space of $3^2 * 4^5 = 9216$, that is 9216 rules in one RL-induced policy. While these types of polices can specify the exact action to take in each case, they are usually too narrow to be aligned to existing learning theories. Each of the rules covers only a very specific case and the relationship between rules is unknown. Thus it is impossible to explain the power of those rules from the perspective of learning theory. The opacity of those induced rules not only hinders us in improving data-driven methodologies when they go wrong, it also prevents us from advancing learning science research more generally. Moreover, it is possible that some of the decisions are environment-specific and may not generalize to other contexts. This in turn prevents translating these induced policies to environments other than the one from which they are induced. Therefore, a general method is needed to shed some light on the extracted detailed data-driven policies.

Decision tree (DT) induction is a robust data mining approach which can be used to extract a compact set of rules from a set of specific examples. It builds a tree-like hierarchical decision-making pattern which represents the knowledge it learned. Each path from root to leaf represents a single rule which may be dealt with separately. Prior studies have shown that DTs can match training examples in most cases, even with relatively small trees. Davidson et al., for example, built a DT for predicting the extinction risk of mammals [6]. Each of the species was described by 11 ecological features (e.g body mass, geographic range and population density) and were labeled with their extinction risk (threatened vs. non-threatened). Their tree contained 20 general rules which covered 4500 training examples, with a decision accuracy over 80%. Additionally, Reinchard et al. built a DT for predicting the invasiveness of woody plants [13]. The resulting DT encoded 15 rules from 235 examples, with a decision accuracy over 76%. Therefore, in our study, we will apply DT to extract general pedagogical decision-making rules from the detailed RL-induced policies.

In short, our primary research question is: *is DT an effective methodology for extracting more general pedagogical rules from the detailed RL-induced pedagogical rules?* In order to investigate this question, we will build DTs using the rules in a RL-induced policy as training examples and empirically evaluate the effectiveness of the extracted set of DT rules by comparing it to the full set of RL-induced rules in a classroom study. The state features in the RL-induced policies are the input features for the DT and the pedagogical actions are the output labels. In our empirical evaluation, we separate the pedagogical decisions from the instructional content, strictly controlling the content so that it is equivalent for all participants by 1) using an ITS which provides equal support for all learners; and 2) focusing on tutorial decisions that cover the same domain content, in this case WE versus PS.

## 2. BACKGROUND

### 2.1 Applying RL to ITSs

Beck et al. applied RL to induce pedagogical policies that would minimize the time students take to complete problems on AnimalWatch, an ITS for grade school arithmetic [2]. They trained the model with simulated students. The low cost of generated data allowed them to apply a model-free RL method, Temporal Difference learning. During the test phase, the induced policies were added to AnimalWatch and the new system was empirically compared with the original system. Their results showed that the policy group spent significantly less time per problem than their no-policy peers. Note that their primary goal was to reduce the amount of time per problem, however faster problem-solving does not always result in better learning performance. Nonetheless, their results showed that RL can be successfully applied to induce pedagogical policies for ITSs.

Iglesias et al., on the other hand, focused on applying RL to improve the effectiveness of an Intelligent Educational System that teaches students DataBase Design [8, 9]. They applied another model-free RL algorithm, Q-learning to induce policies that provide students with direct navigation support through the system's content. They used simulated students to induce the policy and empirically evaluated its effectiveness on real students. Their results showed that while the policy led to more effective system usage behaviors from students, the policy students did not outperform the no-policy peers in terms of learning outcomes.

Shen investigated the impact of both immediate and delayed reward functions on RL-induced policies and empirically evaluated the effectiveness of the induced policies within

an Intelligent Tutoring System called Deep Thought [15]. The induced pedagogical policies are used to decide whether the next task should be WE or PS. They found that some learners benefited significantly more from effective pedagogical policies than others.

Finally, Chi et al. applied model-based RL to induce pedagogical policies to improve the effectiveness of an Intelligent Natural Language Tutoring System for college-level physics called Cordillera [4]. The authors collected an exploratory corpus by training human students on an ITS that makes random decisions and then applied RL to induce pedagogical policies from the corpus. They showed that the induced policies were significantly more effective than the prior ones.

In short, prior studies have shown that RL-induced pedagogical policies can improve students' learning or reduce training time. However, all of these studies focused on the effectiveness of the RL-induced policies. None of them considered extracting more general rules from the induced policies.

## 2.2 Extracting General Rules

In addition to the work of Davidson et al. [6] and Reinchard et al. [13], DTs have been used for other tasks. Vayssiers et al., for example, applied Classification And Regression Trees to predict the presence of 3 species of oak in California [18]. Their training examples were Vegetation Type Map records for 2085 unique locations. Each record consisted of 25 climatic and geographic features as well as 3 labels showing the presence of the species (Quercus agrifolia, Quercus douglasii and Quercus lobata). One DT was induced for each type. The DTs were tested on another dataset which contains the same type of records for 2016 locations. For Quercus agrifolia, the induced tree had 10 leaf nodes and 94.9% of its predictions are correct for the locations that have the presence of this oak (sensitivity) while 86.7% of its predictions are correct for cases without the oak (specificity). For Quercus douglasii, the induced tree had 22 leaf nodes and a sensitivity and specificity of 87% and 79.9% respectively. For Quercus lobata, the tree had 6 leaves but reached a sensitivity of 77% and a specificity of 73.3%.

Thus, prior studies have shown that DT can effectively extract a small set of general decision-making rules from a large set of specific examples. However, all the examples used by these studies were observations of existing phenomena. So far as we know, this work is the only relevant research on the application of DT to extract a compact set of decision-making rules directly from full RL-induced rules and empirically evaluated the two sets of the rules.

## 2.3 Applying DT to RL

Prior research on incorporating DT with RL has largely focused on seeking a better representation of state space or policy for RL. Boutilier et al [3]. proposed representational and computational techniques for Markov Decision Processes (MDPs) to reduce the size of the state space. They used dynamic Bayesian networks and DTs to represent stochastic actions as well as DTs to represent rewards. Based upon this representation, they then developed algorithms to find conditional optimal policies. Their method was empirically evaluated on several planning problems and they showed significant savings in both time and space for some types of problems. Gupta et al. proposed the Policy Tree algorithm for RL. This algorithm is designed to directly induce a functional representation of the conditional optimal policies as a DT. They evaluated it on a variety of domains and showed that it was able to make splits properly [7].

In short, prior researchers have shown that properly combining DT with RL can result in a large amount of savings in time and space for finding good policies. However, none of these studies directly applied DT on RL-induced policies.

# 3. INDUCE FULL SET OF RL-POLICY

Previously, researchers have typically used the Markov Decision Process (MDP) [16] framework to model user-system interactions. The central idea behind this approach is to transform the problem of inducing effective pedagogical policies on what action the agent should take to the problem of computing an optimal policy for an MDP.

## 3.1 Markov Decision Process

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

After defining an MDP, we can transfer the student-system interaction dialog into the trajectory which can then 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 $R_i$ in state $S_i$, and then transferred to the next state $S_{i+1}$. In general, the reward can be divided into two categories, immediate and delayed, where immediate rewards are received during the state transition, and delayed are available after reaching to goal state.

## 3.2 Training Datasets

Our training dataset was collected from three exploratory studies in which students were trained on an ITS which made random yet reasonable pedagogical decisions. The studies were given as homework assignments during CSC226: Discrete Mathematics, a core CS course offered at NCSU during the Fall 2014, Spring 2015 and Fall 2015 semesters. The dataset contains a total of 149 students' interaction logs. All students used the same ITS, followed the same general procedure, studied the same training materials, and worked through the same training problems. In order to model the students' learning process, we extracted a total of 142 state feature variables, which can be grouped into 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 spent 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 Inducing RL Policies

In order to apply RL to induce pedagogical policies, we first defined the pedagogical decision-making problem as an MDP. The state representation includes all of the relevant features available at the beginning of each *step*. The actions are WE and PS at the *step* level. The transition tables were calculated on our training dataset, and our reward function includes two types of reward: delayed and immediate. Our most important reward is based on normalized learning gain (NLG) ($\frac{posttest-pretest}{1-pretest}$), which measures the students' learning gains *irrespective of their incoming competence*. This reward was given as a delayed reward as NLG scores can only be calculated after students finish the entire training process. However, Shen et al. [15] showed that giving immediate rewards can lead to the production of more effective policies when compared to delayed rewards. This is known as the credit-assignment problem. The more that 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 success or failure. Therefore, for the purposes of this study we also assigned immediate rewards based upon the students' performance during training on the system.

The value iteration algorithm was applied to find the optimal policy. This algorithm operates by finding the optimal value for each state $V^*(s)$. The optimal value for a given state is the expected discounted reward that the agent will gain if it starts in $s$ and follows the optimal policy to the goal. Generally speaking, $V^*(s)$ can be obtained by the optimal value function for each state-action pair $Q^*(s,a)$ which is defined as the expected discounted reward the agent will gain if it takes an action $a$ in a state $s$ and follows the optimal policy to the end. The optimal state value $V^*(s)$ and value function $Q^*(s,a)$ can be obtained by iteratively updating $V(s)$ and $Q(s,a)$ via equations 1 and 2 until they converge:

$$Q(s,a) := R(s,a) + \gamma \sum_{s' \in S} p(S_j|S_i,a)V(s') \quad (1)$$

$$V(s) := \max_a Q(s,a) \quad (2)$$

Here, $p(S_j|S_i,a)$ is the estimated transition model $T$, $R(s,a)$ is the estimated reward model and $0 \leq \gamma \leq 1$ is a discount factor.

To induce effective pedagogical policies, we combined RL with various feature selections including 10 types of correlation-based methods and an ensemble method and capped the maximum number of state feature size to be eight. More details of our feature selection methods are described in [14]. The final resulting RL policy involves seven state features and 3706 rules.

# 4. EXTRACTING COMPACT DT-RL SETS

In order to extract a more compact set of decision-making rules from the full set of RL-induced rules, we implemented the ID3 algorithm to build DTs [12]. Each rule in the final RL-induced policy was used as a training example. Two types of decision trees were built: unweighted and weighted, as well as two types of pruning strategies were implemented: pre- and post-pruning. Next, we will discuss each of them in turn.

## 4.1 Unweighted vs. Weighted Tree

The decision to give a WE vs. PS may impact students' learning differently in different situations. We therefore built two types of decision trees: unweighted and weighted. *Unweighted* trees treated each decision equally while *weighted* trees take account of the relative importance of each pedagogical rule. When applying the value iteration algorithm to induce the optimal policy, we generate the optimal value function $Q^*(s,a)$, which gives the expected discounted reward each agent will gain if it takes an action $a$ in a state $s$ and follows the optimal policy to the end. For a given state $s$, a large difference between the values of $Q(s,\text{“}PS\text{”})$ and $Q(s,\text{“}WE\text{”})$ indicates that it is more important for the ITS to follow the optimal decision in the state $s$. We therefore used the absolute difference between the Q values for each state $s$ to weight each RL pedagogical rule.

The ID3 algorithm builds a tree recursively from root to leaves. On each iteration of the construction process the algorithm will check the state of the dataset for the current branch. It will then select a test feature for the current node based upon the weighted information gain. The current node will then be expanded by adding branches to it, each of which represents a possible value for the selected feature. The data will be partitioned over the branches according to the value of the test feature. The selected feature cannot be used again by its children. Weighted information gain is defined by the difference between the weighted entropy of the examples before it is selected and after they are separated by feature value. The weighted entropy of a node can be calculated by equation 3

$$H(G) = -\sum_{i=1}^{J} p(i|G) log_2 p(i|G) \quad (3)$$

$J$ is the total number of output label classes. In our case, it is the number of pedagogical actions (WE or PS) which is 2 . $p(i|G)$ is the weighted frequency defined by the equation: $p(i|G) = \frac{\sum_{x \in i} w_x}{\sum_{y \in G} w_y}$. $\sum_{x \in i} w_x$ is the total weight of the examples which are in node $G$ and which belong to class $i$. And $\sum_{y \in G} w_y$ is the total weights of examples in node $G$.

The information gain of spliting the current set of training examples using feature $F$ can be calculated by equation 4:

$$IG(F,G) = H(G) - \sum_{j=1}^{k} p(t_j|G)H(t_j) \quad (4)$$

$p(t_j|G)$ is the weighted frequency of the examples in node $G$: $p(t_j|G) = \frac{\sum_{x_F=t, x\in G} w_x}{\sum_{y\in G} w_y}$. $\sum_{x_F=t, x\in G} w_x$ is the total weights of examples in nodes $G$ whose value of feature $F$ is $j$ and $\sum_{y\in G} w_y$ is the total weight of examples in nodes $G$.

## 4.2 Pre-Pruning and Post-Pruning

To control the size of rules induced by DT, we examined two types of pruning strategy: pre- and post-pruning. The pre-pruning is conducted during the process of building the tree and it used the information gain to determine whether to expand or to terminate. Only nodes with an information gain greater than a threshold times its depth: $IG(F,G) \geq \theta \times D_G$ will be expanded and others will be made as a leaf. $\theta$ is a fixed threshold and $D_G$ is the depth of node $G$.

Post-Pruning is conducted after the whole decision tree is built and it used the error rate as the pruning measure. The error rate before a node is expanded is defined as: $e_G = \frac{\sum_{i\in I} w_i}{|G|}$. $I$ is the set of the decisions incorrectly classified by node $G$ and $|G|$ is the total number of examples in the node $G$. The error rate after a node is expanded is defined as: $e_C = \frac{\sum_{c\in C}\sum_{j\in I_c} w_i}{|G|}$. $C$ is the set of children nodes of $G$ after it is expanded and $I_c$ is the set of the decisions incorrectly classified by the node $c$. In post-pruning, if the difference of a node's error rate from before to after split is less than a threshold, the node will be pruned by removing all of its branches to make it a leaf node.

## 4.3 The Compact Set of DT-RL Rules

In order to induce a compact set of DT-RL rules, we applied the DTs to the full set of 3706 RL-induced rules. The induced unweighted and weighted DTs without pruning has 2527 and 2456 rules (leaf nodes) respectively. Thus, without pruning, DTs are already able to extract a smaller set of rules: it reduced the total number of rules by over 1000.

Figure 1 shows the relationship between the number of leaf nodes (x-axis) and the inverted weighted accuracy (y-axis). Weighted accuracy($WA$) is the weighted percentage of decisions correctly made, which can be calculated by the equation: $WA = \frac{\sum_{d_i\in T} w_i}{\sum_{d_i} w_i}$. $T$ is the set of correct predictions made by a DT and $w_i$ is the weight of decision $i$. The inverted weighted accuracy ($IWA$) is $IWA = WA^{-10}$, *the lower the better*. Since our goal is to find a good balance point between the IWA and the number of leaf nodes, we applied a widely used strategy called the Elbow Method, to select the best tree. As we can see in the figure, the elbows for the two unweighted tree approaches are around 800 and 1700 rules (x-axis) for the pre and post pruning respectively while the elbows for the two weighted tree approaches are around 250 and 500 for the pre and post pruning respectively. So it seems that weighted tree can extract more compact set of rules than the unweighted trees. While the weighted pre-pruning approach has around 250 rules, its IWA is much higher than the weighted post-pruning approach. Therefore, we chose the weighted tree with post-pruning strategy which has the an elbow at about 500 leaf nodes and reasonable IWA.

To further justify our DT choice, Table 1 shows the relationship between the pruning thresholds, $WA$ and the number

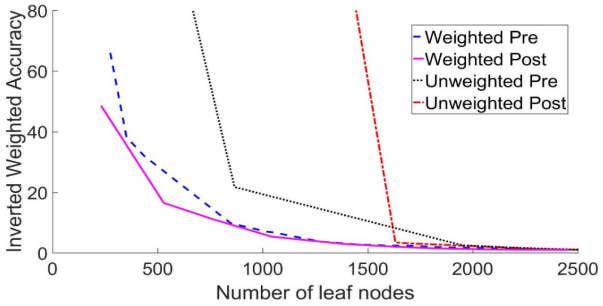


**Figure 1: Leaf Nodes - Accuracy**

of leaf nodes for the weighted tree with post-pruning. Table 1 shows that the tree with the closest number of leaves to 500 is the 529 one. It can be obtained by apply a pruning threshold of 0.8 and the result tree has a weighted accuracy of 0.76. The rules in the resulted tree will be the rules used in the DT-RL condition.

In short, we applied DT on RL-induced pedagogical policies to extract a more compact set of decision-making rules. The effectiveness of the original full set and the compact set of policies were empirically compared against a baseline policy which makes random yet reasonable decisions: PS vs. WE. Thus, we have three conditions:

1. Full-RL: the full set of 3706 RL-induced rules.
2. DT-RL: the compact set of 529 DT-induced RL rules.
3. Random: the random yet reasonable policy.

# 5. EMPIRICAL EXPERIMENT

**Participants:** This study was conducted in the undergraduate Discrete Mathematics course at the Department of Computer Science at NC State University in the Fall of 2016. 153 students participated in this study, which was given as their *final* homework assignment.

**Conditions:** Students in the study were assigned to three conditions via balanced random assignment based upon their course section and performance on the class mid-term exam. Since the primary goal of this work is to examine the effectiveness of the two RL based policies, we assigned more students to the Full-RL and DT-RL conditions than in the random condition. The final group sizes were: $N = 61$ (Full-RL), $N = 51$ (DT-RL), and $N = 41$ (Random).

Due to preparations for exams and length of the experiment, 126 students completed the experiment. 5 students were excluded from the subsequent analysis due to perfect pretest scores, working in group or gaming the system during the training. The remaining 121 students were distributed as follows: $N = 45$ for Full-RL; $N = 41$ for RL-DT; $N = 35$ for Random. We performed a $\chi^2$ test of the relationship between students' condition and their rate of completion and found no significant difference among the conditions: $\chi^2(2) = 0.955, p = 0.620$.

**Probability Tutor:** Pyrenees is a web-based ITS for probability. It covers 10 major principles of probability, such as the Complement Theorem and Bayes' Rule. Pyrenees

**Table 1: Weighted DT with Post-pruning**

| Threshold | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| WA | 1.00 | 0.99 | 0.98 | 0.96 | 0.93 | 0.89 | 0.85 | 0.79 | 0.76 | 0.68 |
| leaves | 2456 | 2217 | 2029 | 1809 | 1608 | 1383 | 1043 | 758 | 529 | 231 |

provides step-by-step instruction and immediate feedback. Pyrenees can also provide on-demand hints prompting the student with what they should do next. As with other systems, help in Pyrenees is provided via a sequence of increasingly specific hints. The last hint in the sequence, the bottom-out hint, tells the student exactly what to do. For the purposes of this study we incorporated three distinct pedagogical decision modes into Pyrenees to match the three conditions.

**Procedure:** In this experiment, students were required to complete 4 phases: 1) pre-training, 2) pre-test, 3) training on Pyrenees, and 4) post-test. During the pre-training phase, all students studied the domain principles through a probability textbook, reviewed some examples, and solved certain training problems. The students then took a pre-test which contained 14 problems. The textbook was not available at this phase and students were not given feedback on their answers, nor were they allowed to go back to earlier questions. This was also true of the post-test.

During phase 3, students in all three conditions received the same 12 rather complicated problems in the same order on Pyrenees. Each main domain principle was applied at least twice. The minimal number of steps needed to solve each training problem ranged from 20 to 50. These steps included defining variables, applying principles, and solving equations. The number of domain principles required to solve each problem ranged from 3 to 11. All of the students could access the corresponding pre-training textbook during this phase. Each step in the problems could have been provided as either a WE or PS based upon the condition policy. Finally, all of the students completed a post-test with 20 problems. 14 of the problems were isomorphic to the pre-test given in phase 2. The remaining six were non-isomorphic complicated problems.

**Grading Criteria:** The test problems required students to derive an answer by writing and solving one or more equations. We used three scoring rubrics: binary, partial credit, and one-point-per-principle. Under the binary rubric, a solution was worth 1 point if it was completely correct or 0 if not. Under the partial credit rubric, each problem score was defined by the proportion of correct principle applications evident in the solution. A student who correctly applied 4 of 5 possible principles would get a score of 0.8. The one-point-per-principle rubric in turn gave a point for each correct principle application. All of the tests were graded in a double-blind manner by a single experienced grader. The results presented below are based upon the partial-credit rubric but the same results hold for the other two. For comparison purposes, all test scores were normalized to the range of [0,1].

# 6. EMPIRICAL RESULTS

Since both the Full-RL and DT-RL policies are based on an RL-induced policy, we combined the two conditions together as the *Induced group* to evaluate the effectiveness the RL-induced policy. The evaluation was conducted by comparing the Induced group with the baseline *Random condition* on learning performance and training time. Moreover, in order to further discover to what extent the compact policy retained the power of the full policy, we compared the Full-RL and DT-RL conditions on the same measures. Next, we will discuss each of the comparisons in turn.

## 6.1 Induced vs. Random

We measured Students' incoming competence via the pre-test scores collected before training took place. Table 2 shows a comparison between the Induced group and the Random group in terms of learning performance. The parenthesized values following the group names in row 1 denote the number of students in each group. The second row in this table shows the pre-test scores. The last column shows the pairwise t-test results. Pairwise t-tests on students' pre-test scores show that there is no significant difference between the two groups: $t(119) = -0.346$, $p = 0.730$, $d = 0.069$. Thus, despite attrition, the two groups remained balanced in terms of incoming competence. Next, we will compare the two groups in terms of learning performance in the post-test and training time.

Rows 2 - 4 in Table 2 show a comparison of the pre-test, isomorphic post-test (14 isomorphic questions), and adjusted post-test scores between the two groups along with the mean and SD for each. In order to examine the students' improvement through training on Pyrenees, we compared their scores on the pre-test and isomorphic post-test questions. A repeated measures analysis using test type (pre-test and isomorphic post-test) as factors and test score as the dependent measure showed a main effect for test type: $F(1, 119) = 98.75$, $p < 0.0001$. Further comparisons on group by group basis showed that on the isomorphic questions, both groups scored significantly higher in the post-test than in the pre-test: $F(1, 85) = 81.30$, $p < 0.0001$ for Induced and $F(1, 34) = 18.30$, $p = 0.0001$ for Random respectively. This suggests that the basic practice and problems, domain exposure, and interactivity of our ITS might help students to learn even when pedagogical decisions are made randomly.

In order to investigate the effectiveness of the induced policies, we compared students' overall learning performance, which was evaluated by their adjusted post-test scores, between the two groups. A one-way ANCOVA analysis was conducted on their overall post-test scores (20 questions), using the pretest scores as a covariate to factor out the influence of their incoming competence. The result shows a significant main effect: $F(1, 118) = 4.628$, $p = 0.033$. That is, the Induced group significantly outperformed the Random group on adjusted post-test scores, which is shown in

**Table 2: Induced vs. Random**

| Cond | *Induced*(86) | *Random*(35) | T-test Result |
|---|---|---|---|
| Pre | .686(.194) | .699(.171) | $t(119) = -0.346$, $p = 0.730$, $d = 0.069$ |
| Iso Post | .851(.155) | .812(.195) | $t(119) = 1.141$, $p = 0.256$, $d = 0.229$ |
| Adjusted Post | .751(.144) | .689(.138) | $t(119) = 2.162$, $p = 0.033$, $d = 0.433$ |
| Time | 105.87(34.30) | 111.18(27.33) | $t(119) = -0.815$, $p = 0.417$, $d = 0.163$ |
| WE steps | 205.74(62.73) | 189.46(11.39) | $t(119) = 1.522$, $p = 0.131$, $d = 0.305$ |
| PS steps | 173.69(61.14) | 190.26(10.28) | $t(119) = -1.591$, $p = 0.114$, $d = 0.319$ |
| WE pct(%) | 54.16(16.35) | 49.89(2.78) | $t(119) = 1.532$, $p = 0.128$, $d = 0.307$ |

the fourth row of Table 2. Therefore, the results showed that the induced policies are significantly more effective than the random policy.

The fifth row in Table 2 shows the average amount of total training time (in minutes) students spent on our ITS for each group. Pairwise t-test showed no significant difference in training time between the two groups: $t(119) = -0.815$, $p = 0.417$, $d = 0.163$. The results suggest that when compared to the random policy, the induced policies generally do not have a significant different impact on students' training time.

The last three rows in Table 2 show the number of WE and PS steps given as well as the percentage of WE steps received by the Induced and the Random group. Pairwise t-tests showed that there is no significant difference between the two groups on these three measures.

## 6.2 Full-RL vs. DT-RL

We then performed the same comparison between the Full-RL and DT-RL conditions in order to examine the effectiveness of the DT-extracted compact policy. The second row in Table 3 shows the pre-test scores for each condition. A pairwise t-test on the scores shows no significant difference between the two conditions: $t(84) = -0.168$, $p = 0.867$, $d = 0.036$. Thus the two conditions were balanced in terms of incoming competence.

The pre-test, isomorphic post-test and adjusted post-test scores are shown in rows 2 - 4 of Table 3. A repeated measures analysis using test type (pre-test and isomorphic post-test) as factors and test score as dependent measure showed a main effect for test type: $F(1, 85) = 81.30$, $p < 0.0001$. Further comparisons on group by group basis showed that both conditions scored significantly higher in isomorphic post-test than in pre-test: $F(1, 44) = 42.16$, $p < 0.0001$ for Full-RL and $F(1, 40) = 39.16$, $p < 0.0001$ for DT-RL. These results suggest that the students can effectively learn from Pyrenees with the full and compact policies.

In order to discover to what degree the compact policy retained the effectiveness of the full policy, we compared the post-test scores between the two conditions. The results of a pairwise t-test showed no significant different between them on isomorphic post-test: $t(84) = 0.505$, $p = 0.615$, $d = 0.109$. We also conducted an ANCOVA analysis on the overall post-test scores using the pretest scores as a covariate and still found no significant different between the two conditions: $F(1, 83) = 0.348$, $p = 0.557$. In short, while on post-test scores, the DT-RL condition scored slightly lower than the Full-RL condition, the difference is not significant.

The fifth row of Table 3 shows the average amount of time students spent on training. As the row shows, the Full-RL condition spent significantly more time than the DT-RL condition: $t(84) = 3.829$, $p = 0.0002$, $d = 0.827$. Thus the Full-RL and DT-RL policies have significant different impact upon the students' training time.

The last three rows of Table 3 show the number of WE and PS steps given and the percentage of WE steps received by the Full-RL and the DT-RL condition. Pairwise t-tests showed that comparing to the DT-RL condition, the Full-RL condition received significantly fewer WE steps: $t(84) = -4.952$, $p < 0.0001$, $d = 1.069$; received a lower percentage of WE steps: $t(84) = -4.955$, $p < 0.0001$, $d = 1.070$; and completed more PS steps: $t(84) = 4.999$, $p < 0.0001$, $d = 1.079$. These results suggest that the pedagogical decisions made by the compact and full policies are substantively different.

# 7. DISCUSSION

In this study, we applied DT to extract a compact set of pedagogical rules from the full set of RL-induced rules and empirically evaluated the effectiveness of two sets of rules in a classroom study. Our goal was to shed some light on the RL-induced policies and we think this is only the first step towards narrowing the gap and building a bridge between machine-induced pedagogical policies and learning theories.

In order to find the best DT, we explored two types of tree: unweighted and weighted; and for each of them, we conducted two types of pruning strategy: pre- and post-pruning. After comparing the performance among them, we selected the weighted tree with the post-pruning strategy to perform the extraction of general decision-making rules. The RL-induced policy contains 3706 specific rules, and the compact DT-RL consisted of 529 rules with a weighted decision accuracy of 76%.

In our empirical experiment, we were able to strictly control the domain content and thus to isolate the impact of *pedagogy* from *content*. Based on this isolation, we compared students' performance with the Full-RL policy, the DT-RL policy and the baseline random policy. Our results showed that students in all three conditions learned significantly after training on Pyrenees, this suggests that the basic training of the ITS is effective, even when the pedagogical decisions are made randomly. To evaluate the effectiveness of the two machine induced policies (Full-RL policy and DT-RL policy), we combined the Full-RL and DT-RL condition as the Induced group and compared its learning performance with the Random group. Our results showed that the Induced

**Table 3: Full-RL vs. DT-RL**

| Cond | Full-RL(45) | DT-RL (41) | T-test Result |
|---|---|---|---|
| Pre | .683(.205) | .690(.184) | $t(84) = -0.168,\ p = 0.867,\ d = 0.036$ |
| Iso Post | .859(.145) | .842(.168) | $t(84) = 0.505,\ p = 0.615,\ d = 0.109$ |
| Adjusted Post | .757(.144) | .739(.145) | $t(84) = 0.594,\ p = 0.554,\ d = 0.128$ |
| Time | 118.42(35.000) | 92.10(27.95) | $t(84) = 3.829,\ p = 0.0002,\ d = 0.827$ |
| WE steps | 177.44(48.86) | 236.80(62.03) | $t(84) = -4.952,\ p < 0.0001,\ d = 1.069$ |
| PS steps | 201.47(47.22) | 143.20(60.57) | $t(84) = 4.999,\ p < 0.0001,\ d = 1.079$ |
| WE pct(%) | 46.77(12.78) | 62.26(16.13) | $t(84) = -4.955,\ p < 0.0001,\ d = 1.070$ |

group significantly outperform the Random group. These results suggest that the machine induced policies are indeed more effective than the random policy.

Finally, in order to examine to what extent the compact DT-RL policy retained the power of the full RL-induced policy, we compared the learning performance of the Full-RL and the DT-RL conditions. Our results suggest that while some of the power was lost in the general rules extraction, the relative performance difference between the Full-RL and the DT-RL condition is not significant. In addition, our results on the pedagogical decisions made in training revealed that the compact DT-RL policy selected significant more WE than the Full-RL policy. This suggests that the two sets of policies indeed made materially different decisions. However, since the weighted DT took account of the importance of each rule, the DT-RL policy aims to retain maximal decision effectiveness from the Full-RL policy while the size of the former is less than 15% of the size of the Full-RL rules. In the future, we will apply existing learning theories to the decision-making process generated by decision tree to find a theoretical basis for the DT-induced general pedagogical decision-making rules.

## 8. ACKNOWLEDGEMENTS

This research was supported by the NSF Grant #1432156: "Educational Data Mining for Individualized Instruction in STEM Learning Environments" and #1651909: "Improving Adaptive Decision Making in Interactive Learning Environments".

## 9. REFERENCES

[1] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.

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

[3] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1):49–107, 2000.

[4] 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.

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

[6] A. D. Davidson and et al. Multiple ecological pathways to extinction in mammals. *Proceedings of the National Academy of Sciences*, 106(26):10702–10705, 2009.

[7] U. D. Gupta, E. Talvitie, and M. Bowling. Policy tree: Adaptive representation for policy gradient. In *AAAI*, pages 2547–2553, 2015.

[8] 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(1):89–106, 2009.

[9] 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.

[10] K. R. Koedinger and et al. Intelligent tutoring goes to school in the big city. *IJAIED*, 8(1):30–43, 1997.

[11] P. Phobun and J. Vicheanpanya. Adaptive intelligent tutoring systems for e-learning systems. *Procedia-Social and Behavioral Sciences*, 2(2):4064–4069, 2010.

[12] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[13] S. H. Reichard and C. W. Hamilton. Predicting invasions of woody plants introduced into north america. *Conservation Biology*, 11(1):193–203, 1997.

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

[15] S. Shen and M. Chi. Reinforcement learning: the sooner the better, or the later the better? In *UMAP*, pages 37–44. ACM, 2016.

[16] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[17] K. Vanlehn. The behavior of tutoring systems. *IJAIED*, 16(3):227–265, 2006.

[18] M. P. Vayssières, R. E. Plant, and B. H. Allen-Diaz. Classification trees: An alternative non-parametric approach for predicting species distributions. *Journal of vegetation science*, 11(5):679–694, 2000.