Student Subtyping via EM-Inverse Reinforcement Learning

Xi Yang[1], Guojing Zhou[1], Michelle Taub[2], Roger Azevedo[2], Min Chi[1]

[1] North Carolina State University, Raleigh, NC 27606, USA
{yxi2, gzhou3, mchi}@ncsu.edu
[2] University of Central Florida, Orlando, FL 32816, USA
{michelle.taub, roger.azevedo}@ucf.edu

ABSTRACT

In the learning sciences, heterogeneity among students usually leads to different learning strategies or patterns and may require different types of instructional interventions. Therefore, it is important to investigate student subtyping, which is to group students into subtypes based on their learning patterns. Subtyping from complex student learning processes is often challenging because of the information heterogeneity and temporal dynamics. Various inverse reinforcement learning (IRL) algorithms have been successfully employed in many domains for inducing policies from the trajectories and recently has been applied for analyzing students' temporal logs to identify their domain knowledge patterns. IRL was originally designed to model the data by assuming that all trajectories have a *single* pattern or strategy. Due to the heterogeneity among students, their strategies can vary greatly and the design of traditional IRL may lead to suboptimal performance. In this paper, we applied a novel expectation-maximization IRL (EM-IRL) to extract heterogeneous learning strategies from sequential data collected from three simulation environments and real-world longitudinal students' logs. Experiments on simulation environments showed that EM-IRL can successfully identify different policies from the heterogeneous sequences with different strategies. Furthermore, experimental results from our educational dataset showed that EM-IRL can be used to obtain different student subtypes: a *"learning-oriented"* subtype who learned the material as much as possible regardless of the time in that they spent significantly more time than the other two subtypes and learned significantly; an *"efficient-oriented"* subtype who learned efficiently in that they not only learned significantly but also spent less time than the first subtype; a *"no learning"* subtype who spent less amount of time than first subtype and failed to learn.

Keywords

Subtyping, learning progression modeling, Student strategy, Inverse reinforcement learning

1. INTRODUCTION

With the rapid development of educational technologies, longitudinal students' learning progression trajectories are readily available. It is often challenging to analyze large-scale heterogeneous progression trajectories to infer high-level information embedded in student subgroups. This challenge motivates the development of student modeling [1, 2, 3, 4] and instructional intervention [5, 6, 7, 8].

Student subtyping, which seeks student groups with similar learning progression trajectories, is crucial to address the heterogeneity in the students, which ultimately leads to personalized instruction where students are provided with interventions tailored to their unique learning status. Student subtyping facilitates the investigation of different types of pedagogical strategies. From the data mining perspective, student subtyping is posed as an unsupervised clustering task of grouping students according to their historical records. Since these records are longitudinal and interrelated, it is important to capture the dependencies among the elements of the recorded sequence to learn more effective and robust representations, which can be utilized in the clustering stage to obtain the student subgroups.

This work aims at investigating *student subtyping* based on their pedagogical strategies, which can be seen as a process of self-regulated learning [9, 10, 11, 12, 13] by setting one's learning goals and ensuring the goals to be attained. Specifically, we focus on students' pedagogical decision-making strategies during their interactions with an intelligent tutoring system (ITS) to learn the probability. In this ITS, once a problem is presented, the students will decide whether they want the ITS *to tell* them how to solve the next problem or complete the next step, by presenting a worked example, or they want the ITS *to elicit* the next problem or take the next step themselves, by requiring problem solving. When making pedagogical decisions, the students have to self-regulate their own learning process which may change the learning outcomes even though the instructional content is controlled. We believe that students' pedagogical strategies are closely related to metacognition, i.e., the processes involved in thinking about thinking [14].

Reinforcement learning (RL) offers one of the most promising approaches to induce effective pedagogical strategies directly from data. A number of researchers have studied applying RL to improve the effectiveness of ITSs, e.g. [15, 7, 16, 17, 18, 19, 20, 8, 21, 22], and much of the prior work fo-

cused on inducing effective policies that determine the best action for the ***ITS*** to take in any given situation so as to maximize a cumulative reward, which is often the student learning gain. On the other hand, in this work, our goal is to infer ***students'*** pedagogical strategies based on their behaviors and decisions while interacting with the ITS.

To do so, we applied *inverse reinforcement learning (IRL)*. Unlike RL, where the reward function is explicitly given as input, IRL takes a bunch of trajectories as input and from which a reward function will be inferred. Given this inferred reward function, the RL can be further deployed to induce the decision-making policy. Since the students' decisions are generally made based on a trade-off among various complex factors, e.g., time, learning gain, difficulty of problems, etc., merely taking the learning gain as the reward cannot reflect the actual decision-making patterns. As a result, we employed IRL to learn students' strategies based on their behavioural data. Recently, IRL has been widely employed in various domains to understand how decisions are made in the given trajectories [23, 24]. Specifically, it has been employed in educational domains to analyze students' temporal log data to identify their domain knowledge patterns [25, 26]. However, IRL was originally designed to model the data by assuming that all trajectories share a *single* pattern or strategy. Considering the heterogeneity among students, their pedagogical strategies can vary greatly and the design of traditional IRL may lead to suboptimal performance. Though we can apply IRL individually for each student, it will forfeit our goal of revealing some general and meaningful patterns from students' trajectories in consideration of the heterogeneity among subgroups of students.

We employed a novel expectation–maximization IRL (EM-IRL) algorithm [27] to model the heterogeneity among student subtypes by assuming that different student subtypes have different pedagogical strategies and students within each subtype share the same strategy. The EM-IRL would recursively cluster students into different subgroups and induce a policy for each group by IRL until both clusters and policies get converged. In the original EM-IRL work, it requires the number of clusters to be pre-defined [27]. However, when applying it to student subtyping in education, it is often hard to figure out beforehand how many types of strategies are involved in students' trajectories. Therefore, we embedded the original EM-IRL into a general framework which can automatically determine the optimal number of clusters from the data.

In this work, we evaluated our general framework on three simulation environments: Grid World, Highway, and Mountain Car, and on real-world longitudinal students' logs collected from an ITS. Our results in three simulation environments showed that EM-IRL could accurately cluster the data with different decision-making strategies. In addition, the experimental results showed that EM-IRL could be easily employed to obtain the student subtypes. Specifically, we got three student subtypes: a *"learning-oriented"* subtype who try to learn the material as much as possible regardless of the time spent and they learned significantly from pre- to post-test; an *"efficient-oriented "* subtype who learn efficiently in that they not only learned significantly but also spent significantly less time than the first subtype; a *"no learning"* subtype who spent the less time and failed to learn. The clustering results suggested the potential of targeting the students who are not using effective pedagogical strategies, adapting the interventions, and offering the students effective pedagogical skill training through the ITS.

The remaining parts are organized as follows. In Section 2, related works are reviewed. Section 3 presents the methods, including the RL, IRL, and EM-IRL. Section 4 displays preliminary results we got in three simulation environments. Section 5 details data collected from the ITS. In Section 6, we discuss the experimental setup for EM-IRL and some other clustering methods. Section 7 presents the experimental results. Finally, Section 8 summarizes the paper.

2. RELATED WORKS

2.1 Students' Subtyping

Previous research has widely explored modeling of student subtyping to assist teachers in providing more targeted interventions at the right time. Generally, student subtyping was analyzed via unsupervised clustering methods. For example, Lopez et al. employed an expectation maximisation clustering method to determine if the students' participation in course Moodle forum could be a good predictor of the final marks [28]. Durairaj and Vijitha applied K-means clustering to predict the pass/fail percentage of the students who appeared for a particular examination [29]. Khalil and Ebner clustered the students into appropriate categories based on their level of engagement [30], so that the teachers could increase retention and improve interventions for specific sub-population. All of these methods were based on the static data, without considering the dynamic properties during learning.

With the rapid development of e-learning, an increasing amount of sequential data was collected via ITSs. In general, the clustering methods to handle sequential data could be generalized into three categories: proximity-based, feature-based, and model-based [31]. More specifically, proximity-based methods measures the similarity between the pair-wise data via the distance calculated by the longest common subsequence, dynamic time warping, etc. For example, Shen and Chi proposed a temporal clustering framework which measured pair-wise distance between the students by dynamic time warping and then clustered them by hierarchical clustering [32]. Their method identified some distinctive patterns among the clusters, which could provide benefits to the personalized learning. Feature-based approaches would first compress the sequential data to be static, then the clustering methods taking static data as input could be further employed. For example, in [33] and [34], the authors aggregated the students' activities to a feature vector and then applied K-means clustering to recognize learner groups in exploratory learning environments. In the model-based methods, the similarity of two data could be calculated based on the likelihood of one of them given the model derived from the other. For example, Li and Yoo proposed to use a Markov chain based clustering methodology to model the students' online learning behaviors collected during the learning process for more effective and adaptive teaching [31]. Additionally, Kock and Paramythis proposed a method combining K-means clustering with dis-

crete Markov models to identify new, semantically meaningful problem-solving styles of the learners [35].

2.2 Students' Pedagogical Strategies

A number of researchers have investigated students' pedagogical decision-making [36, 37, 38, 39, 40, 41]. Previous research has shown that students make pedagogical decisions strategically. For example, Aleven et al. conducted a study to investigate students' hint usage behavior [36]. Results showed that students used the easy-to-apply intelligent help more often than the Glossary. However, students often waited long before asking for a hint. When requesting hints, they often skipped the intermediate hints to reach the bottom-out hint which showed the solution directly. The results suggested that students preferred less effort-taking help (intelligent help and bottom-out hint), and oftentimes, they used the help less than they needed.

Additionally, prior research showed that providing students with pedagogical decision-making assistance could result in better decision-making skills or learning performance. Roll et al. [37] examined the relationship between students' help-seeking patterns and learning performance. They found that asking for help on challenging steps was generally productive while help abusing behaviors were correlated with poor learning. Mitrovic et al. [38] compared three types of decision-making modes: system control, student control, and faded control. Under the faded control, the system selected the problem for the student at the beginning of the training and gave explanations of why the problems should be selected. As the training proceeded, the control was given to the students. Results showed that the faded control group demonstrated improved problem selection skill and achieved better learning gain than the other two groups. Long et al. [39] compared an assistance condition, where problem selection assistance was provided, with standard condition (no assistance). Their results showed the assistance condition achieved significantly better learning performance and better declarative knowledge of a key problem-selection strategy comparing to the standard condition.

2.3 Learning From Demonstrations

Learning from demonstrations [42], also known as imitation learning [43] or apprenticeship learning [44], is a process to reproduce the decision-making behaviors in demonstrated trajectories. Generally, the methods in this area can be categorized into two groups: 1) directly learning a policy as a state-action mapping by parroting the demonstrated behaviors, which is typically done via supervised learning; and 2) inferring rewards from the demonstrations and then applying reinforcement learning (RL) to induce the policy, which is called inverse reinforcement learning (IRL). The latter is generally preferred because the reward is a more robust, succinct, and transferable definition for a task [45]. Specifically, comparing to supervised learning, IRL has higher generalization ability to robustly learn from smaller size trajectories collected from larger state spaces, and the succinctly represented reward function can be handily transferred to other agents in different scenarios.

Based on how the rewards are inferred, existing IRL algorithms can be generalized into two categories: maximum margin-based methods and probabilistic model-based methods. Specifically, maximum margin-based methods infer rewards by finding a model to maximizes the margin between the demonstrated trajectories and other alternative behaviors [44]. However, it is often suffers from the ill-posed issue with non-uniqueness [45], i.e., there can be multiple reward functions to explain the demonstrated behaviors. Probabilistic model-based methods, on the other hand, are able to handle this issue by using probability distributions to introduce preferences over reward functions [46]. In this category, Ramachandran and Amir [47] proposed a Bayesian IRL, which combined prior knowledge and evidence from the demonstrated trajectories to derive a probability distribution over the reward functions. Similarly, Ziebart et al. proposed a maximum entropy IRL which results in the least biased estimation of the reward function [23]. Babes-Vroman et al. [27] proposed a maximum likelihood IRL (MLIRL), which finds the reward function that maximize the probability to observe the demonstrated behaviors. Their experimental results showed that the MLIRL outperformed some other IRL methods, including the linear programming based maximum margin IRL and maximum entropy IRL.

All the above methods assume a *single* reward function for all demonstrations. Some other approaches have been proposed to handle the *multiple* reward functions. Dimitrakakis and Rothkopf [48] proposed a Bayesian multi-task IRL, which learns a reward function for each individual trajectory using the same prior distribution. Choi et al. [49] proposed a method based on nonparametric Bayesian IRL in which the prior of mixing distribution of different rewards was modeled by the Dirichlet process. Babes-Vroman et al. [27] proposed an EM-based framework, which iteratively computes the probabilities that the demonstrations belong to each cluster and updates the cluster-wise rewards based on MLIRL. Considering the efficiency and good performance EM-based method, we adapted it for analysis in this work.

Recently, IRL has been widely applied in various domains. Ziebart et al. [23] employed it in driver route modeling for predicting driving behaviors as well as for route recommendation. Asoh et al. [24] applied IRL to medical records and explored the potential rules in doctors' diagnosis. Of most relevance, IRL also showed effectiveness in educational domain. Rafferty et al. applied IRL in education applications to automatically infer learners' beliefs in an education game [25]. They demonstrated that IRL could recover the participant's beliefs towards how their actions could affect the environment, which indicated the potential to utilize IRL to interpret data from interactive educational environments. Then in another of their work, IRL was further employed to assess learners' mastery of some skills in solving algebraic equations. Based on the learned IRL results, some skills the learners misunderstood could be detected and personalized feedback for improving the skills were further rendered [26].

3. METHOD

3.1 Reinforcement Learning

Markov decision process (MDP) was widely utilized to model the user-system interactions. The central idea behind reinforcement learning (RL) is to transform the problem of inducing effective policies into a computational problem of finding an optimal policy for choosing actions in MDP. An

MDP describes a stochastic control process using a 5-tuple $< S, A, T, R, \gamma >$. Taking the pedagogical policy induction as an example, S indicates the learning environment states, which is often represented by student-system interaction features. A denotes the tutor's possible actions, such as elicit or tell. The reward function R is generally assigned as students' learning performance. The transition probability T can be estimated from training data. $\gamma \in [0, 1)$ denotes a discount factor for the future rewards. Given a defined MDP, we can transform our student-system interaction logs into trajectories as: $s_1 \xrightarrow{a_1,r_1} s_2 \xrightarrow{a_2,r_2} \cdots s_n \xrightarrow{a_n,r_n}$. Here $s_i \xrightarrow{a_i,r_i} s_{i+1}$ indicates that at the i^{th} turn, the learning environment was in state s_i; the tutor executed action a_i and received reward r_i; then the environment transferred into the state s_{i+1}.

In traditional RL, the reward function R serves as a guidance to praise or punish the agent's behaviors to fulfil a certain task when interacting with the environment. Therefore, it is essential and needs to be elaborately hand-crafted in advance to reflect the task. In ITS, the reward is generally formulated as the students' learning performance, e.g., learning gains, since the intention of tutor's decision-making is to promote students' learning. However, the reward function in students' decision-making is more complex to be determined: students may have various learning patterns, e.g., finishing the process as quick as possible or working hard regardless of the time, which is cumbersome to be manually encoded in a reward function. The different reward functions reflected the different strategies students employed during the training process. Therefore, if student's reward function can be learned in a data-driven manner, we can better understand their pedagogical decision-making strategies.

3.2 Inverse Reinforcement Learning

3.2.1 General IRL

The difficulty of the reward function design triggered the development of the inverse reinforcement learning (IRL). IRL follows a reverse procedure comparing to the traditional RL: in RL, given the reward function, the agent will learn an optimal policy; while in IRL, the trajectories derived from the optimal policy are given, from which the agent will learn the reward function. It can be described as a stochastic control process using a 4-tuple $MDP\backslash R =< S, A, T, \gamma >$ where the reward function is missing.

In general framework of IRL, the input is a $MDP\backslash R$ together with some demonstrated trajectories $\mathcal{T}$. The reward function $\mathcal{R}_\theta$ parameterized by θ can be modeled as either a linearly weighted sum of feature values or belonging to a certain distribution. Most of the existing IRL methods follow 3 steps: in step 1, the parameter θ is randomly initialized; in step 2, given the $\mathcal{R}_\theta$, general RL methods can be applied to induce the policy; In step 3, the divergence of the behaviors regarding to the learned policy and the given trajectories is minimized to update the θ. The step 2 and step 3 are repeated until the divergence is reduced to a desired level.

To investigate students' pedagogical strategy, we can feed their decision-making trajectories into the IRL model. Once the reward function is learned, the strategy can be further induced via traditional RL methods. Herein, we compared some most commonly utilized IRL methods including: quadratic programming based maximum margin IRL [44], maximum entropy IRL [23], Bayesian IRL [47], and maximum likelihood IRL (MLIRL) [27] over three online simulation environments (i.e., Grid World, Highway, and Mountain Car). We found MLIRL always outperformed others and it is also most time-efficient. As a result, we take MLIRL for the IRL-based analysis hereinafter.

General Process of IRL
Input $MDP\backslash R =< S, A, T, \gamma >$ and trajectories $\mathcal{T}$ **Output** $\mathcal{R}_\theta$
step 1 Initialize the parameter θ in reward function **step 2** Solve the MDP to learn the policy π **step 3** Update the optimization θ to minimize the divergence between $\mathcal{T}$ and behaviors following the π **Repeat step 2** and **step 3** until convergence

3.2.2 Maximum Likelihood IRL

To formally define the maximum likelihood IRL, we denote the input N demonstrated trajectories as $\mathcal{T} = \{\xi_1, ..., \xi_N\}$ and each trajectory is composed of a set of state-action pairs: $\xi_i = \{(s_1, a_1), (s_2, a_2), ...\}$. The reward function is defined as the linear function of feature vector for state-action pairs: $r_\theta(s, a) = \theta^T \phi(s, a)$. Then the Q-value can be calculated as:

$$Q_\theta(s, a) = \theta^T \phi(s, a) + \gamma \sum_{s'} T(s, a, s') \bigotimes_{a'} Q_\theta(s', a'), \quad (1)$$

$$where \ \bigotimes_{a} Q_\theta(s, a) = \frac{\sum_a Q_\theta(s, a) \exp(\beta Q_\theta(s, a))}{\sum_{a'} \exp(\beta Q_\theta(s, a'))} \quad (2)$$

Eq. 2 shows the Boltzmann exploration. Comparing to standard Bellman equation, it enables the likelihood to be differentiable, thus the objective function can be easier optimized. β represents the degree of confidence and it is set as 0.5 in our experiments. The Boltzmann exploration policy parameterized by θ is:

$$\pi_\theta(s, a) = \frac{\exp(\beta Q_\theta(s, a))}{\sum_{a'} \exp(\beta Q_\theta(s, a'))} \quad (3)$$

Then the log-likelihood of trajectories $\mathcal{T}$ is calculated as:

$$L(\mathcal{T}|\theta) = log \prod_{i=1}^{N} \prod_{(s,a)\in\xi_i} \pi_\theta(s, a)^{\omega_i} = \sum_{i=1}^{N} \sum_{(s,a)\in\xi_i} \omega_i log \pi_\theta(s, a) \quad (4)$$

Herein, ω_i denote the weight for ξ_i, which can be estimated by its frequency of the occurrence. By maximizing the Eq. 4, the parameter θ enables the trajectories $\mathcal{T}$ to have highest probability to be observed given the reward function R_θ. Once the reward function is learned, the strategy followed by $\mathcal{T}$ can be further induced by any RL method, e.g., policy iteration that we employed in this work.

In general, IRL methods assume the reward function to be unique for all input trajectories. However, it is often the case that the trajectories are heterogeneous and have various reward functions. For example, in ITS, students' decision-making behaviors can have different patterns which cannot be easily captured by a single IRL model. As a result, a model suitable for multiple reward functions is favored.

Algorithm: MLIRL

Input $MDP\backslash R$, trajectories $\mathcal{T}$, trajectories' weights ω_i, $i = 1, ..., N$, learning rate α
Initialize reward parameter θ randomly
Repeat
  Learn the policy π_θ
  Compute $L = \sum_i \sum_{(s,a)\in\xi_i} \omega_i log(\pi_\theta(s,a))$
  Update $\theta = \theta + \alpha \bigtriangledown L$
  Until target number of iterations completed

3.3 Expectation–maximization IRL

To deal with trajectories with multiple reward functions, i.e., multiple strategies, Babes-Vroman et al. [27] proposed a straight-forward expectation–maximization IRL (EM-IRL). Herein, we adapted the original EM-IRL to automatically determine the optimal number of clusters. Instead of directly assigning the cluster number, we considered a possibly maximal number of clusters, i.e., K_{max}, and a variable k initialized as 2 indicating the current cluster number.

Specifically, to determine the optimal number of clusters, starting from the cluster number $k = 2$, we iteratively implemented the EM procedure, until a pre-defined *stop_criteria* was met. The *stop_criteria* was defined as: either there were some empty clusters generated or the log-likelihood (LL) of the clustering results defined in Eq. 5 varied smaller than a pre-defined threshold comparing to the last iteration, which we set as 10. The LL reflected the clustering performance by measuring the accordance of learned clusters with the correspondingly induced cluster-wise strategies. In Eq. 5, N_j stands for the number of trajectories in cluster j.

$$LL = \sum_{j=1}^{k} \sum_{i=1}^{N_j} log(z_{ij}) \tag{5}$$

$$z_{ij} = Pr(\xi_i|\theta_j) = \prod_{(s,a)\in\xi_i} \frac{\pi_{\theta_j}(s,a)\rho_j}{Z}, \tag{6}$$

Before the EM loop, parameters ρ_j and θ_j, $j = 1, .., k$, which denoted the estimated prior probability and reward parameter for the j^{th} cluster were randomly initialized.

In the ***E step***, the probability that trajectory i belongs to cluster j was calculated by Eq.6, in which Z is a normalization factor; In the ***M step***, the prior probability of cluster is updated by Eq. 7. Meanwhile, the reward parameter θ_j can be learned by any IRL and herein we employed the MLIRL with weights of trajectories being z_{ij}.

$$\rho_j = \sum_i \frac{z_{ij}}{N} \tag{7}$$

The E step and M step will be iteratively executed until a target number of iterations is completed, which was set as 80 in this work to ensure the convergence. Finally, we found k clusters when LL got converged, with each cluster standing for a group of trajectories with an unique reward function. Based on these reward functions, we could further induce the cluster-wise strategies.

4. SIMULATION ENVIRONMENTS

Algorithm: EM-IRL

Input $MDP\backslash R$, trajectories $\mathcal{T}$, maximal number of clusters K_{max}
Initialize $k = 2$
While $k \leqslant K_{max}$
  Initialize ρ_j and θ_j, $j = 1, ..., k$, randomly
  Repeat
    E Step: Compute the z_{ij}, $i = 1, ..., N$
    M Step: Update the prior probability ρ_j; and
      Learn reward parameter θ_j via MLIRL
    Until target number of iterations completed
  If *stop_criteria* is True: **Break**; **Else**: $k = k + 1$

Since the ground-truth of students' subtypes were unknown in advance, it is difficult to directly evaluate the EM-IRL learned clusters from the students' data. Thus, we first carried out EM-IRL in three simulation environments which had decided ground-truth. If different strategies could be accurately distinguished by EM-IRL in simulations, we would be more confident to further deploy it in ITS environment.

4.1 Environment Settings

We explored three simulation environments including Grid World, Highway, and Mountain Car, as shown in Figure 1.

Grid World: adapted from [27], in which three grids were randomly chosen as puddles indicated by bricks in Figure 1(a).

- **States (25)** 5×5 grid-size.
- **Actions (4)** Moving to up, down, left, or right.
- **Strategies (3)** Moving to the 1) upper-right corner; 2) lower-left corner; or 3) lower-right corner.

The rewards are designed for the three strategies: 1) Upper-right corner has the reward of 10; 2) Lower-left corner has the reward of 10; 3) Lower-right corner has the reward of 10. Otherwise, each state was punished -1.

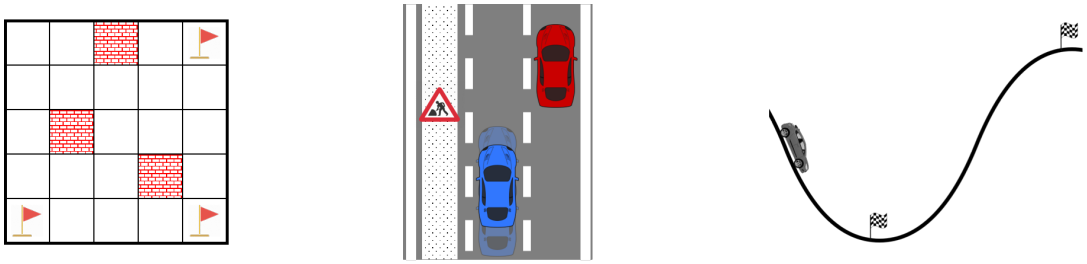
Highway: adapted from a three-lane highway scenario introduced in [50], in which the agent controlled a blue car with three speed levels, which could switch between the three lanes or go off-road on either side. At all timestamps, there would be a red car in one of the three lanes.

- **States (729)** the blue car's speed had 3 levels and could move horizontally in 9 locations; the red car could move vertically in 9 locations and horizontally in 3 locations.
- **Actions (5)** Staying at the current state, speeding up, slowing down, moving left, or moving right.
- **Strategies (2)** 1) Keeping off the left lane (suppose it is under construction); 2) Driving at the fastest speed.

The rewards are designed for the two strategies: 1) Driving on the left lane has the reward of -10; 2) Driving with the lowest level of speed has the reward of -10. In both strategies, off-road is punished -0.5, collision is punished -5, and maintaining the state has no reward.

Mountain Car: adapted from the MountainCar-v0 in OpenAI Gym [51], in which a car was on a one-dimensional track and moves between two mountains.

- **States (80)** 10 horizontal positions with 8 levels of speed.



(a) Grid World (b) Highway (c) Mountain Car

Figure 1: Three simulation environments: (a) Grid World; (b) Highway; (c) Mountain Car.

Table 1: Cluster-wise and overall purities by EM-IRL clustering in three simulation environments.

Environment	Cluster-wise		Overall
	Strategy Idx	Purity (%)	Purity (%)
Grid World	1	100	100
	2	100	
	3	100	
Highway	1	100	100
	2	100	
Mountain Car	1	100	96.4
	2	93.2	

- **Actions (3)** Pushing left, no pushing, or pushing right.
- **Strategies (2)** 1) Reaching to the right mountain top (the car needs to drive back and forth to build up enough momentum to push up); 2) Parking at the valley bottom.

The rewards are generated for the two strategies: 1) Right mountaintop has the reward of +10; 2) Valley bottom has the reward of +10. Otherwise, each state is punished -1.

In each environment, the initial states were randomly assigned, the transitions between states were stochastic and estimated from the data. For each strategy, we induced a policy via policy iteration and employed it to collect trajectories. Specifically, the number of collected trajectories for each strategy was 500, 1000, and 1000 in three environments, respectively. In each environment, trajectories with various strategies were mixed together and fed into the EM-IRL.

Given the ground-truth of cluster-belongings in simulation environments, the results of EM-IRL were evaluated by the purity of each cluster and across overall clusters. Denote the size of i^{th} cluster as N_i with ground-truth labels $\mathbf{L}_i$, then the cluster-wise purity is calculated as the number of majority labels divided by the cluster size, i.e., $purity_i = \frac{majority(\mathbf{L}_i)}{N_i}$; and the overall purity is calculated by the mean of purity among all clusters, i.e., $purity = \frac{1}{k}\sum_{i=1}^{k} purity_i$.

4.2 EM-IRL Results in Three Simulations

The EM-IRL clustering results for the three simulation environments are shown in Table 1, in which the first column is the environment; second and third columns show the index of strategy and the corresponding cluster-wise purity; the last column show the overall purity among all clusters.

In Grid World, all strategies could be accurately clustered by EM-IRL. Specifically, both cluster-wise purities and overall purity were 100%. Likewise, in Highway, the two strategies were accurately clustered with the purity of 100%. In Mountain Car, a few trajectories of driving to right mountaintop (Strategy 0) were mis-clustered to the parking at valley (Strategy 1). This is because the mis-clustered trajectories tried to move to left to collect enough momentum, which showed very similar behaviors to reaching the valley. Overall, the results suggested the effectiveness of EM-IRL in accurately distinguishing subtypes of trajectories with different strategies in all three simulation environments.

5. ITS LEARNING ENVIRONMENT

Our data was collected by letting students work on a web-based ITS, which taught college students probability, e.g., Addition Theorem and Bayes' Theorem. The instruction was conducted by guiding students go through training problems. For each problem, the tutor provided step-by-step instruction, immediate feedback, and on-demand help. The help was provided via a sequence of increasingly specific hints. The last hint in the sequence, i.e., the bottom-out hint, told the student exactly what to do. During training, the students could make pedagogical decisions on whether to solve the next step by themselves or observe the tutor to solve it. If they choose to solve by themselves, the tutor will *elicit* the solution from them by asking questions; otherwise, the tutor will show or *tell* them the solution directly.

5.1 Data Collection

All students participating in our data collection went through four phases: textbook, pre-test, training, and post-test. During *textbook*, all students studied the domain principles from a probability textbook. They read a general description of each principle, reviewed some examples of it, and solved some single- and multiple-principle problems. Then the students took a *pre-test* which contained 14 problems. During this phase, they would not be given feedback on their answers, nor be allowed to go back to earlier questions (this was also true for the post-test). During the *ITS training* procedure, students received 12 problems in the same order. 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. Such steps included variable definitions, principle applications, and equation solving. The

number of domain principles required to solve each problem ranged from 3 to 11. Finally, all students took the *post-test* which contained 20 problems in total. 14 of the problems were isomorphic to the problems given in the pre-test phase, while the remaining 6 were harder non-isomorphic multiple-principle problems.

The pre- and post-tests 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 we presented were 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 $[0, 100]$.

We measure students' learning performance using normalized learning gain (NLG), which measured their gain *irrespective of their incoming competence*. It is calculated as: $NLG = \frac{post-pre}{100-pre}$, where pre and $post$ refer to the students' test scores before and after the ITS training respectively and 100 is the maximum score. Herein, for the post-test, we considered all 20 problems that are either isomorphic and non-isomorphic. In addition, an isomorphic NLG (Iso_NLG) was also measured. Unlike NLG, the Iso_NLG was calculated based on the pre- and isomorphic post-test scores, which contained only 14 isomorphic multiple-principle problems.

5.2 States & Actions

Our dataset contains 127 students. Each student spent ~ 2 hours on the system and completed around 400 steps.

States 142 state features were extracted from the student-system interaction log data. Specifically, the features can be grouped into five categories:

- **Autonomy** (10 features): the amount of work done by a student, such as the number of elicits since the last tell;
- **Temporal** (29): time related information about the student's behavior, such as the average time per step;
- **Problem Solving** (35): information about the current problem solving context, such as problem difficulty;
- **Performance** (57): information about the student's performance so far, such as the percentage of correct entries;
- **Hints** (11): information about the student's hint usage, such as the total number of hints requested.

For each category, we employed K-means clustering to get the discretized states. By selecting an elbow of errors when the clustering results got converged, the number of states for each category of features was determined as follows: Autonomy (3 states), Temporal (4), Problem Solving (3), Performance (4), and Hints (3). As a result, we got 432 discrete states totally. Based on the discretized states, we estimated the transition probabilities from all available data.

Actions The students can take two action of elicit/tell, i.e., to *elicit* the solution by themselves through asking questions, or to let the tutor *tell* them the solution directly.

6. EXPERIMENTAL SETTINGS

6.1 Student Subtyping by EM-IRL

Based on the EM-IRL learned clusters, we conducted analyses by checking the statistical significance among different clusters' learning performance, including the pre-test scores, isomorphic NLG (Iso_NLG), NLG, students' learning time on the training task (Time), and the percentage of elicit in students' decisions (Elicit_Perc).

6.2 Student Subtyping by Other Methods

6.2.1 Clustering by Traditional Methods

To evaluate the clustering performance of EM-IRL, we compared it with three other clustering methods: two K-means based approaches that took the pre-test scores and the learning state in the final step as the input respectively and a K-medoids based approach that took dynamic time warping (DTW) [52] distance between trajectories as the input. The K-means based approaches were static-information-based clustering while the K-medoids based DTW considered dynamic state transitions in the trajectories. In our experiments, each of these methods generated three clusters and for each cluster, the MLIRL was employed to learn a strategy. Based on the learned strategies, we calculated the log-likelihood (LL, referring to Eq. 5) of observing such clustering results.

6.2.2 Clustering by Matching RL / IRL Policies

We further explored whether RL or IRL policies could model the heterogeneity in student decision-makings. The inducing of these two policies are detailed as follows.

Inducing the RL policy: To investigate whether students' learning strategies could be distinguished from the tutor's perspective, we compared students' decisions to a RL induced pedagogical policy and clustered the students based on the matching rate. Since the RL policy was induced with the goal of improving students' learning performance, it is expected that the group with a higher matching rate with the RL policy would have better learning performance.

Specifically, we applied RL to learn a pedagogical policy that determines whether the next step should be elicit or tell (the same decisions students made in our ITS). The training data set contained 1,118 students' interaction logs collected from a series of seven prior studies which followed the identical procedure and learning materials as the students in this study described in Section 5. The same 142 features used by EM-IRL were extracted from the logs and used to induce the policy. In an empirical classroom study, the policy was compared with a deep Q-network (DQN) induced policy and a random policy. Results showed that the RL policy significantly outperformed both of them [21].

Once the RL policy was induced, we applied it on the student decision-making data (127 students) to see what decision the RL policy would make on each step. Then, we calculated the matching rate between students' decisions and the RL policy individually for each student. Based on the matching

rates, the students were split into three groups via K-means clustering, denoted as High, Medium, or Low based on the average matching rate of the group.

Inducing the IRL Policy: Similarly, to investigate whether students' learning strategies could be distinguished from their own perspective, we applied IRL to induce a policy from student decision-making data and compared students' decisions with the IRL policy. Given that our data analysis showed that most of students learned significantly from ITS training, herein, we assumed that a majority of students completed the training with the goal to learn. Thus, we expected that the group with a higher matching rate with the the IRL policy would have better learning performance.

The IRL policy was induced from the 127 students who were given the opportunities to make pedagogical decision during training. Herein, the MLIRL algorithm [27] was utilized for policy induction. Similar to the RL based method, the IRL policy was applied back to students' data to calculate the matching rate between students' decisions and the IRL policy. Then, K-means clustering was applied on the matching rate to cluster students into High, Medium, or Low groups.

7. RESULTS

7.1 Student Subtyping by EM-IRL

Fitting students' data to the EM-IRL framework in Section 3.3, when *stop_criteria* was met, we got three clusters. Table 2 shows the EM-IRL subtyping results. From left to right, it shows the students' subtypes, number of students (# Stu), pre-test score (Pre), isomorphic NLG (Iso_NLG), NLG, time on the training task (Time), and percentage of elicit in students' decisions (Elicit_Perc). Based on statistical analysis, we named the three resulting clusters as: *learning-oriented*, *efficient-oriented*, and *no learning*.

A one-way ANOVA analysis on pre-test scores showed no significant difference among the three clusters: $F(2, 124) = 1.36$, $p = 0.260$, $\eta = 0.022$. This suggested that students in the three clusters were balanced in incoming competence. To measure students' learning gain in training, we conduced analyses on their Iso_NLG and NLG. A one-way ANOVA analysis on Iso_NLG showed a significant difference among the three clusters: $F(2, 124) = 3.24$, $p = 0.042$, $\eta = 0.050$. Subsequent contrast analysis revealed that *learning-oriented* > *no learning*: $t(124) = 2.54$, $p = 0.012$, $d = 0.75$ and *efficient-oriented* > *no learning*: $t(124) = 2.19$, $p = 0.030$, $d = 0.54$. Similar results were found for NLG in that a one-way ANOVA analysis showed a significant difference among the three clusters: $F(2, 124) = 3.73$, $p = 0.027$, $\eta = 0.057$. Subsequent contrast analysis revealed that *learning-oriented* and *efficient-oriented* significantly outperformed *no learning*: $t(124) = 2.73$, $p = 0.007$, $d = 0.77$ and $t(124) = 2.15$, $p = 0.033$, $d = 0.52$ respectively.

In terms of time on task, a one-way ANOVA analysis showed a significant difference among the three clusters: $F(2, 124) = 5.81$, $p = 0.004$, $\eta = 0.086$. Subsequent contrast analysis indicated that *learning-oriented* took longer time on task than the other two clusters: $t(124) = -3.11$, $p = 0.002$, $d = 0.58$ for *efficient-oriented* and $t(124) = 2.37$, $p = 0.019$, $d = 0.63$ for *no learning*. A contrast analysis on the percentage of elicit in students' decisions revealed that *learning-oriented* took significantly more elicit actions than *no learning*: $t(124) = 2.24$, $p = 0.027$, $d = 0.70$.

To summarize, the *learning-oriented* subtype spent significantly more time than the other two groups on the training task and achieved the best performance on both Iso_NLG and NLG (signifiantly higher than *no learning*). This suggested that learning-oriented students mainly focused on learning the materials, regardless of the time they may spend. The *efficient-oriented* subtype significantly outperformed *no learning* on learning performance and at the same time spent significantly less time than *learning-oriented*. This suggested that *efficient-oriented* students could balance learning gain and time on task. Finally, the *no learning* subtype achieved the lowest learning outcomes.

7.2 Student Subtyping by Other Methods

7.2.1 Clustering by Traditional Methods

We compared our EM-IRL with three traditional baseline clustering methods, namely K-means on the pre-test score (K-means on Pre); K-means on the learning state (142 features) in the final step (K-means on Final Step); K-medoids on the DTW distance among trajectories [52], which is calculated based on the 142 features (K-medoids on DTW). The results are shown in Table 3, with the two columns being clustering method and the resulting log-likelihood (LL).

Overall, results showed that the dynamic-information-based clustering approaches (K-medoids on DTW and EM-IRL) performed better than static-information-based approaches (K-means on Pre and K-means on Final Step). Between the two static-information-based approaches, K-means on final Step performed better than K-means on pre-test. This is not surprising because the state in the final step included information generated during training while the pre-test score only included information till the end of pre-test. Between the two dynamic-information-based approaches, EM-IRL outperformed K-medoids on DTW. A possible reason is that EM-IRL took both states and actions into account while K-medoids on DTW considered only the states in trajectories.

7.2.2 Clustering by Matching RL / IRL Policies

Results of Matching with the RL Policy: Based on the matching rate with the RL policy, we got three clusters by K-means: High ($M = .84, SD = .05$), Medium ($M = .70, SD = .05$), and Low ($M = .52, SD = .07$). A one-way ANOVA analysis over the matching rate showed a significant difference: $F(2, 124) = 339.87$, $p < 0.0001$, $\eta = 0.846$. Subsequent contrast analysis showed that: High > Medium: $t(124) = 4.38$, $p < 0.0001$, $d = 0.99$ and Medium > Low: $t(124) = 8.01$, $p < 0.0001$, $d = 1.70$.

A one-way ANOVA analysis on pre-test showed there was no significant difference among the three groups: $F(2, 124) = 0.26$, $p = 0.771$, $\eta = 0.004$. Analyses on Iso_NLG (calculated based on pre-test and isomorphic post-test) and NLG (calculated based on pre-test and full post-test, which contains six additional hard problems) also showed no significant difference among the three groups. In terms of time on the training task, there was a significant difference among the three groups: High ($M = 2.40, SD = .50$), Medium ($M = 2.42, SD = .66$), and Low ($M = 1.88, SD = .40$).

Table 2: EM-IRL clustering results in ITS environment.

Subtype	#Stu	Pre	Iso_NLG	NLG	Time	Elicit_Perc (%)
learning-oriented	50	73.9(16.8)	55.9(45.3)	23.4(53.6)	2.52(.70)	87.53(13.40)
efficient-oriented	64	76.2(14.5)	43.9(92.4)	-4.4(127.2)	2.18(.45)	84.93(15.02)
no learning	13	81.9(17.4)	-21.1(212.1)	-98.4(340.4)	2.10(.50)	77.06(20.04)

Table 3: Comparison of the log-likelihood (LL) for different clustering methods

Method	LL ($\times 10^3$)
K-means on Pre	-10.68
K-means on Final Step	-9.60
K-medoids on DTW	-8.83
EM-IRL	**-6.36**

A one-way AVONA on time shows: $F(2, 124) = 9.21$, $p = 0.0002$, $\eta = 0.129$. Subsequent contrast analysis revealed that the High and Medium groups spent significantly more time than the Low group: $t(124) = 3.85$, $p = 0.0002$, $d = 1.11$ and $t(124) = 3.99$, $p = 0.0001$, $d = 0.92$, respectively. An analysis on the percentage of elicit in students' decisions showed a significant difference among the three groups: $F(2, 124) = 66.97$, $p < 0.0001$, $\eta = 0.519$. Subsequent contrast analysis revealed that High > Medium: $t(124) = 4.38$, $p < 0.0001$, $d = 0.99$ and Medium > Low: $t(124) = 8.01$, $p < 0.0001$, $d = 1.70$.

The results showed that by matching with the RL strategy, we could differentiate students' time-consuming strategies from time-efficient strategies. However, it was not able to identify the student subtypes that made a difference in the learning performance. This suggested the presence of a gap between tutor's and students' strategies. Specifically, comparing to taking actions following the tutor's decisions passively, the students might prefer actively direct their own learning process. Therefore, when deploying the tutor's strategy to students, it might not promote the learning performance as expected.

Results of Matching with the IRL Policy: Based on the matching rate with the IRL policy, we got three clusters by K-means: High ($M = .86, SD = .05$), Medium ($M = .71, SD = .05$), and Low ($M = .54, SD = .06$). A one-way ANOVA analysis over the matching rate showed a significant difference among the three groups: $F(2, 124) = 360.99$, $p < 0.0001$, $\eta = 0.853$. Subsequent contrast analysis showed that: High > Medium: $t(124) = 15.92$, $p < 0.0001$, $d = 3.37$ and Medium > Low: $t(124) = 13.52$, $p < 0.0001$, $d = 3.23$.

A one-way ANOVA analysis on pre-test showed there was no significant difference among the three groups: $F(2, 124) = 1.17$, $p = 0.314$, $\eta = 0.019$. Analyses on the Iso_NLG and NLG also showed no significant difference among the three groups. In terms of time on the training task, there was a significant difference among the three groups: High ($M = 2.44, SD = .54$), Medium ($M = 2.27, SD = .68$), and Low ($M = 2.08, SD = .42$). A one-way AVONA on time shows: $F(2, 124) = 3.11$, $p = 0.048$, $\eta = 0.048$. Subsequent contrast analysis showed that the High group spent significantly more time than the Low group: $t(124) = 2.43$, $p = 0.017$, $d = 0.70$. An analysis on the percentage of elicit in students' decisions showed a significant difference among the three groups: $F(2, 124) = 93.92$, $p < 0.0001$, $\eta = 0.602$. Subsequent contrast analysis revealed that High > Medium: $t(124) = 7.95$, $p < 0.0001$, $d = 1.83$ and Medium > Low: $t(124) = 7.08$, $p < 0.0001$, $d = 1.43$.

The results showed that IRL based policy matching was able to cluster the students' strategies different in time. However, it was unable to learn specific subtype of students whose strategy will lead to better learning outcomes. One possible reason that the IRL-based analyses could not identify the learning-performance-impactful strategies is that a single policy was insufficient to effectively generalize the decision-making patterns for the overall students. Different students might follow heterogeneous decision-making strategies.

In summary, the results suggested that EM-IRL could effectively conduct student subtyping reflecting different decision-making strategies. As a contrast, clustering by traditional methods or by matching RL/IRL policies could not find desired student subtypes.

8. CONCLUSIONS

In this paper, we investigated students' subtyping via EM-IRL. By analyzing students' subtyping, we aimed at putting ourselves in the shoes of students to better understand their decision-making. To evaluate the performance of EM-IRL, we first applied it to three simulation environments, where the EM-IRL displayed robust performance to accurately cluster the trajectories with different strategies. Given the accurate clustering results in simulators, we were more confident to further apply EM-IRL to real world longitudinal students' logs collected from an ITS. The results suggested that the EM-IRL could effectively group students with different subtypes, e.g., learning-oriented, efficient-oriented, and no-learning. As a contrast, clustering by traditional methods or by matching RL/IRL policies could not find desired subtypes. The subtyping results showed the potential of providing tutors evidence to give more customized interventions to better assist students' learning. In the future, we will conduct early clustering to detect students' strategies as early as possible. Besides, empirical studies will be carried out to evaluate the effectiveness of subtyping-based interventions to improve the targeted group of students.

Acknowledgements: This research was supported by the NSF Grants: Generalizing Data-Driven Technologies to Improve Individualized STEM Instruction by Intelligent Tutors(2013502); CAREER: Improving Adaptive Decision Making in Interactive Learning Environments(1651909); Integrated Data-driven Technologies for Individualized Instruc-

tion in STEM Learning Environments(1726550); MetaDash: A Teacher Dashboard Informed by Real-Time Multichannel Self-Regulated Learning Data(1660878).

9. REFERENCES

[1] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[2] Vincent AWMM Aleven and Kenneth R Koedinger. An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive science*, 26(2):147–179, 2002.

[3] Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers*, 36(2):180–192, 2004.

[4] Wilson J González-Espada and Daniel 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*.

[5] Ana Iglesias, Paloma Martínez, Ricardo Aler, and Fernando Fernández. Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems*, 22(4):266–270, 2009.

[6] Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. Faster teaching via pomdp planning. *Cognitive science*, pages 1290–1332, 2016.

[7] Min Chi, Kurt VanLehn, Diane Litman, and Pamela 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.

[8] Guojing Zhou, Jianxun Wang, Collin Lynch, and Min Chi. Towards closing the loop: Bridging machine-induced pedagogical policies to learning theories. In *EDM*, 2017.

[9] Roger Azevedo, Nicholas V Mudrick, Michelle Taub, and Amanda E Bradbury. Self-regulation in computer-assisted learning systems. 2019.

[10] Philip H Winne and Allyson F Hadwin. Studying as self-regulated learning. In *Metacognition in educational theory and practice, The educational psychology series*. 1998.

[11] Philip H Winne and Allyson F Hadwin. The weave of motivation and self-regulated learning. In *Motivation and self-regulated learning*, pages 309–326. 2012.

[12] Jeffrey Alan Greene and Roger Azevedo. A theoretical review of winne and hadwin's model of self-regulated learning: New perspectives and directions. *Review of educational research*, 77(3):334–372, 2007.

[13] Claire Ellen Weinstein, Jenefer Husman, and Douglas R Dierking. Self-regulation interventions with a focus on learning strategies. In *Handbook of self-regulation*, pages 727–747. Elsevier, 2000.

[14] Michelle Taub, Nicholas V Mudrick, and Roger Azevedo. Strategies for designing advanced learning technologies to foster self-regulated learning. *Strategies for deep learning with digital technology: Theories and practices in education*, pages 137–170, 2017.

[15] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach. *International Journal of Artificial Intelligence in Education*, 21(1-2):83–113, 2011.

[16] Shayan Doroudi, Kenneth Holstein, Vincent Aleven, and Emma Brunskill. Towards understanding how to leverage sense-making, induction and refinement, and fluency to improve robust learning. *International Educational Data Mining Society*, 2015.

[17] Kenneth R Koedinger, Emma Brunskill, Ryan SJd Baker, Elizabeth A McLaughlin, and John Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

[18] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1077–1084, 2014.

[19] Jonathan P Rowe and James C Lester. Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework. In *AIED*, pages 419–428. Springer, 2015.

[20] Shitian Shen and Min Chi. Aim low: Correlation-based feature selection for model-based reinforcement learning. *International Educational Data Mining Society*, 2016.

[21] Guojing Zhou, Hamoon Azizsoltani, Markel Sanz Ausin, Tiffany Barnes, and Min Chi. Hierarchical reinforcement learning for pedagogical policy induction. In *International conference on artificial intelligence in education*, pages 544–556. Springer, 2019.

[22] Guojing Zhou, Xi Yang, Hamoon Azizsoltani, Tiffany Barnes, and Min Chi. Improving student-tutor interaction through data-driven explanation of hierarchical reinforcement induced pedagogical policies. In *Proceedings of the 28th Conference on User Modeling, Adaptation and Personalization*. ACM, 2020.

[23] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.

[24] Hideki Asoh, Masanori Shiro1 Shotaro Akaho, Toshihiro Kamishima, Koiti Hasida, Eiji Aramaki, and Takahide Kohro. An application of inverse reinforcement learning to medical records of diabetes treatment. In *ECMLPKDD2013 workshop on reinforcement learning with generalized feedback*, 2013.

[25] Anna N Rafferty, Michelle M LaMar, and Thomas L Griffiths. Inferring learners' knowledge from their actions. *Cognitive Science*, 39(3):584–618, 2015.

[26] Anna N Rafferty, Rachel Jansen, and Thomas L Griffiths. Using inverse planning for personalized feedback. *EDM*, 16:472–477, 2016.

[27] Monica Babes, Vukosi Marivate, Kaushik Subramanian, and Michael L Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning*, pages 897–904, 2011.

[28] Manuel Ignacio Lopez, Jm M Luna, C Romero, and S Ventura. Classification via clustering for predicting final marks based on student participation in forums. *International Educational Data Mining Society*, 2012.

[29] M Durairaj and C Vijitha. Educational data mining for prediction of student performance using clustering algorithms. *International Journal of Computer Science and Information Technologies*, 5(4):5987–5991, 2014.

[30] Mohammad Khalil and Martin Ebner. Clustering patterns of engagement in massive open online courses (moocs): the use of learning analytics to reveal student categories. *Journal of Computing in Higher Education*, 29(1):114–132, 2017.

[31] Cen Li and Jungsoon Yoo. Modeling student online learning using clustering. In *Proceedings of the 44th annual Southeast regional conference*.

[32] Shitian Shen and Min Chi. Clustering student sequential trajectories using dynamic time warping. *International Educational Data Mining Society*, 2017.

[33] Saleema Amershi and Cristina Conati. Automatic recognition of learner groups in exploratory learning environments. In *International Conference on ITS*, pages 463–472. Springer, 2006.

[34] Saleema Amershi and Cristina Conati. Combining unsupervised and supervised classification to build user models for exploratory. *JEDM| Journal of Educational Data Mining*, 1(1):18–71, 2009.

[35] Mirjam Köck and Alexandros Paramythis. Activity sequence modelling and dynamic clustering for personalized e-learning. *User Modeling and User-Adapted Interaction*, 21(1-2):51–97, 2011.

[36] Vincent Aleven and Kenneth R Koedinger. Limitations of student control: Do students know when they need help? In *International conference on ITS*, pages 292–303. Springer, 2000.

[37] Ido Roll, Ryan SJ d Baker, Vincent Aleven, and Kenneth R Koedinger. On the benefits of seeking (and avoiding) help in online problem-solving environments. *Journal of the Learning Sciences*, 23(4):537–560, 2014.

[38] Antonija Mitrovic and Brent Martin. Scaffolding and fading problem selection in sql-tutor. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education*, pages 479–481, 2003.

[39] Yanjin Long and Vincent Aleven. Mastery-oriented shared student/system control over problem selection in a linear equation tutor. In *International conference on intelligent tutoring systems*, pages 90–100. Springer, 2016.

[40] Guojing Zhou, Collin Lynch, Thomas W Price, Tiffany Barnes, and Min Chi. The impact of granularity on the effectiveness of students' pedagogical decisions. In *CogSci*, pages 2801–2806, 2016.

[41] Guojing Zhou and Min Chi. The impact of decision agency & granularity on aptitude treatment interaction in tutoring. In *CogSci*, pages 3652–3657, 2017.

[42] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[43] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[44] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[45] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *ICML*, volume 1, page 2, 2000.

[46] Shao Zhifei and Er Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, 2012.

[47] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[48] Christos Dimitrakakis and Constantin A Rothkopf. Bayesian multitask inverse reinforcement learning. In *European workshop on reinforcement learning*, pages 273–284. Springer, 2011.

[49] Jaedeug Choi and Kee-Eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, pages 305–313, 2012.

[50] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse reinforcement learning through structured classification. In *Advances in NIPS*, pages 1007–1015, 2012.

[51] Praveen Palanisamy. *Hands-On Intelligent Agents with OpenAI Gym: Your guide to developing AI agents using deep reinforcement learning*. Packt Publishing Ltd, 2018.

[52] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.