2017-01-18

Tutoring Students with Adaptive Strategies

Hao Wan
Worcester Polytechnic Institute

TUTORING STUDENTS WITH ADAPTIVE STRATEGIES

By

Hao Wan

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Computer Science

December 2016

APPROVED:

Professor Joseph E. Beck Worcester Polytechnic Institute Advisor	Professor Ivon Arroyo Worcester Polytechnic Institute Committee Member
Professor Neil T. Heffernan Worcester Polytechnic Institute Committee Member	Dr. Bror Saxberg Kaplan, Inc. Committee Member
Professor Craig Wills Worcester Polytechnic Institute Head of Department	

Abstract

Adaptive learning is a crucial part in intelligent tutoring systems. It provides students with appropriate tutoring interventions, based on students' characteristics, status, and other related features, in order to optimize their learning outcomes. It is required to determine students' knowledge level or learning progress, based on which it then uses proper techniques to choose the optimal interventions. In this dissertation work, I focus on these aspects related to the process in adaptive learning: student modeling, k-armed bandits, and contextual bandits.

Student modeling. The main objective of student modeling is to develop cognitive models of students, including modeling content skills and knowledge about learning. Students usually learn skills in sequence since preliminary skills need to be learned prior to the complex skills. However, few research works have utilized this relation in the student models. In this work, I determined the impact of how student performance on prerequisite skills influences ability to learn post-requisite skills. I found a strong gradient with respect to knowledge of prerequisites: students in the bottom 20% of prerequisite knowledge exhibited wheel spinning behavior 50% of the time, while those in the top 20% of pre-required knowledge exhibited wheel spinning behavior only 10% of the time. I also incorporated the prerequisite performance into the wheel spinning model, and it turned out that it was a significant reliable predictor in the model.

K-armed bandits. A k-armed bandit algorithm focuses on selecting an action, in order to maximize total rewards over all time steps. Due to the lack of diverse interventions and small difference of intervention effectiveness in educational experiments, do k-armed bandit algorithms improve students' learning outcomes? In this dissertation work, I proposed a simple selection strategy based on statistical t-tests, call Strawman, and compared it with several k-armed bandit algorithms. The results showed that the Strawman's performance is competitive and it exhibited different exploration/exploitation patterns.

Contextual bandits. In contextual bandit problem, additional side information, also called context, can be used to determine which action to select. We first separated a data set into several groups with a student feature, and then apply a bandit algorithm in each group. The results demonstrated that being combined with the feature could improve rewards of bandit algorithms. Consequently,

another question arose, how to combine multiple context with bandits? A simple solution is to apply a k-armed bandit algorithm in every possible combination of context values. However, its complexity increases exponentially with respect to the number of available features. In this work, I proposed a decision tree algorithm, which is capable of detecting aptitude treatment effect for students. By applying a bandit algorithm in each leaf node of the tree, I evaluate effect of the algorithm in two different types of data sets, simulated data and real experimental data. In the simulated data, the decision tree algorithm finely captured the pre-defined structure, and thus the contextual bandits performed reliably better than the bandit without any context. In the 22 ASSISTments experiments, the effect of context differed, the contextual bandits performed significantly better than the bandits without context in some experiments, while the two strategies are closed to each other in other experiments.

Acknowledgements

I am most grateful to my dissertation advisor, Professor Joseph Beck, for his guidance and support throughout of my dissertation research and Ph.D. studying period. He always refined my work with great patience, and his talent in tacking complex research questions always gave me an insight when I got stuck. Even sometime his jump mind was hard to follow, it always ignited "sparks" in my research work.

I sincerely thank Professor Carolina Ruiz, who served as my first academic advisor and reader of my qualifier examination. She guided me to the right way to do research and helped me to build fundamental skills of being a Ph.D.

I would like to thank Professor Neil Heffernan and Professor Ivon Arroyo for providing me valuable feedbacks in my research, and their problems in the comprehensive examination helped me to obtain broader and deeper knowledge about the research area in my dissertation.

I also want to thank Dr. Bror Saxberg, whose experience in industry brought me fresh air out of the "ivory tower".

I would like to thank everyone in EDMRG team, for the insightful discussions and helpful suggestions. In addition, my thanks go to Chiying Wang, Lei Cao, Kaiyu Zhao, Chuan Lei, and Di Yang for their help in my Ph.D. study.

I want to thank my parents, Qingyun Wan and Dongxiang Wu, for their patience and confidence on me in the past years. They were doing their best to support me studying in USA. I also thank my wife, Meixia Xiong. She quit the job in China and stayed abroad with me to support my studying. I very much appreciate her sacrifice. Finally, my thanks to my two little daughters, Casey Wan and Mingshu Wan, you brought brightness and happiness in my Ph.D. life.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1. Intelligent Tutoring Systems

Intelligent Tutoring Systems (ITSs) are computer programs that are designed to incorporate artificial intelligence techniques to tutor students with customized instructions and feedbacks (Nwana, 1990). The goal of all ITSs is to facilitate students' learning with AI techniques. To support a student's learning, an ITS should know how he/she learns, what to learn, and when to learn. According to (Polson & Richardson, 2013), ITSs consists of four parts:

- *The expert module* defines the representation of knowledge of the experts in the specific domain, and the rules how to convey the knowledge to students. The expert knowledge serves as the source of knowledge to be provided to students, and also provides an evaluation standard to assess students' learning progress (Nwana, 1990).
- *The student model module* informs systems the dynamic stage of students' knowledge and skill. It provides functions that predict, evaluate, and diagnose the students' knowledge and tutorial actions (Self, 1988).
- *The curriculum instruction module* establishes a set of teaching instructions for students. It supplies a student appropriate with pedagogic interactions according to the student's knowledge from student model module and the previously set tutorial goal structure, e.g. hint, tutorial videos and web pages.
- *The user interface module* is a component that enables learners to interact with the systems.

1.2. ASSISTments

The ASSISTments is a web-based ITS which assists and assesses students' learning automatically (Feng, Heffernan, & Koedinger, 2009). It has been widely used in studying Math, English, and other subjects by thousands of Middle and High school students. It allows teachers or curriculum designers to form structure of a problem set with one or more following components:

- Pre-test: this part usually contains a series of problems that are used to warm up students or detect students' initial knowledge, in order to provide appropriate tutorial strategies to students.
- Skill builder: questions in this set are based on a specific skill, and students are required to answer 3 (as default) questions correct in a row to complete the part.

- Complete-all: this problem set, unlike skill builder, requires students to answer all questions.
- Post-test: students are tested with this set of questions after they have completed the skill builder or complete-all part, in order to check how well they have learned in this assignment.

The ASSISTments also provides different forms of assistances, and enables teachers to decide which is conveyed to students in the problems:

- Scaffolding questions: this assistance breaks the original question down into several fundamental steps. As shown in Figure 1, there are two questions; the bottom one is a scaffolding question of the top one. After get all scaffolding questions correctly step by step, the student is then back to answer the original question.
- Hints: a hint provides some clues and suggestions to help students to answer the questions. For example, in the bottom question of Figure 1, by clicking the "show hint" button, the system presents a multiplication table to students.
- Videos and web pages: this assistance provides a link to a tutorial video or web page. It does not focus on helping students to solve a particular question, but tutors students with domain knowledge of related skills.

Furthermore, the ASSISTments system supports researchers to do controlled experiments. That is students are assigned into problem sets with different conditions, e.g. different hints or feedbacks, according to particular rules, and their activities and outcomes are compared to obtain the tutorial efficacy of conditions in the learning progress. For example, Ostrow and Heffernan assess the effects of two different feedback forms, bland texts and videos, in a randomized controlled trial. Their results suggest that students prefer video feedback to bland text, and the video feedback improves students' learning outcomes (Ostrow & Heffernan, 2014).

1.3. Over-tutoring and Under-tutoring

Example 1.

A student is learning a skill through a skill builder in ASSISTments, and he is required to answer 3 questions correctly in a row to master the skill. However, he is lack of prepared knowledge that is necessary, and the questions along with the tutorial feedbacks are hard for him to understand. Consequently, the student cannot master the skill no matter how many opportunities he has tried.

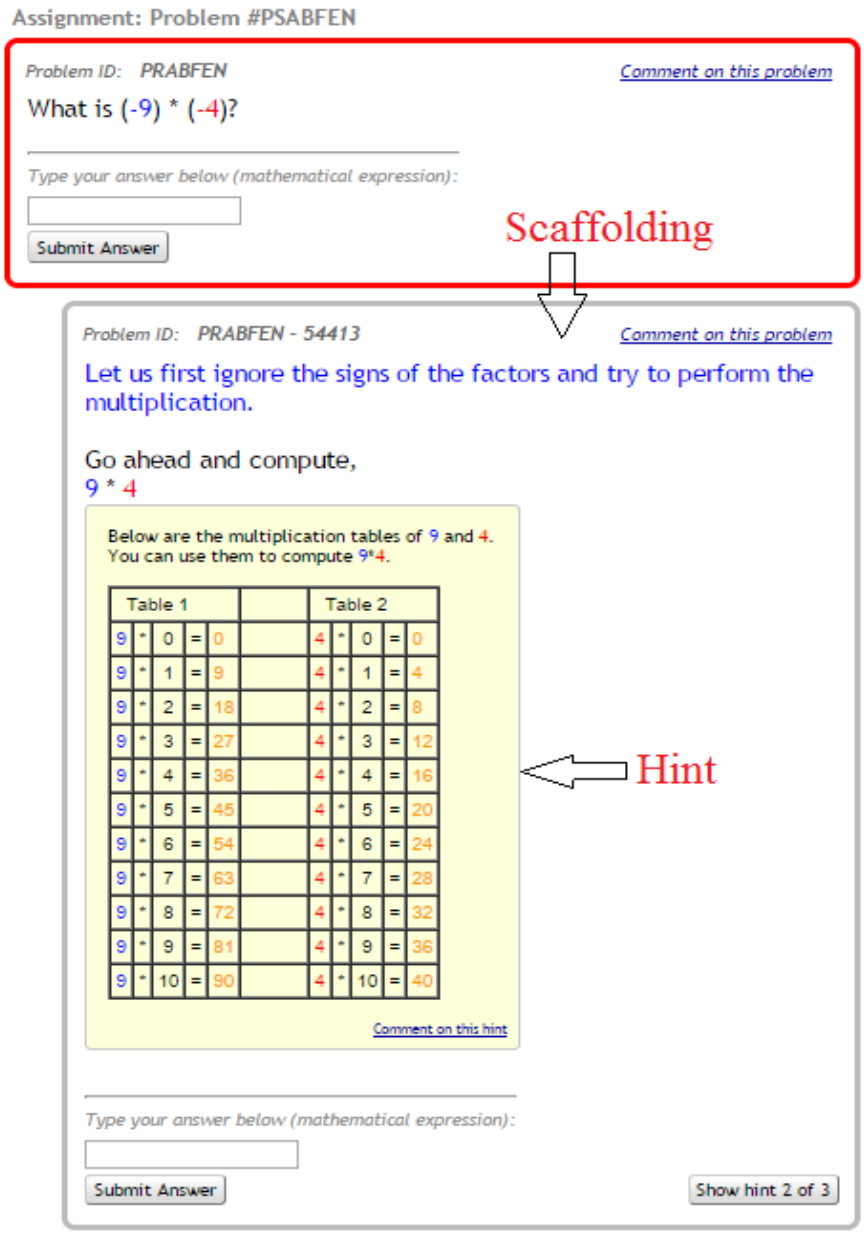


Figure 1. Questions in the ASSISTments. The assistance of the top question is to provide scaffolding questions, while the assistance of the bottom question is to provide hints.

Example 2.

Another student is also learning a skill through a skill builder in ASSISTments. This student has some prior knowledge, and he masters the skill in a few trials. But in the later test, he performs very poorly. It seems that the tutor strategy does not work on him effectively.

When teachers are designing didactic schemas for a class of students, they are targeting the "average" student, because it is impossible for them to develop different didactic sequences and meanwhile investigate which sequence is optimal for each student in a very limit budget of time

(Lopes, Clement, Roy, & Oudeyer, 2013). This can lead to that some students are trained with inappropriate strategies, and thus their outcomes are not as expected. For example, in Example 1, the student is over-tutored with the questions that are far beyond the level of the student's knowledge; while in Example 2; the student is under-tutored with too easy questions. In both of those two examples, the pedagogical processes are failed in tutoring the students.

A variation of didactic schema is to require students to learn a skill by practicing related problems, based on the assumption that students will achieve mastery with enough practices, in order to reach the teaching goal of treating different students with different strategies, such that top students will practice less, while bottom students will practice more. However, Beck and Gong (Beck & Gong, 2013) found that practicing more indeed helps some students to achieve mastery, but the proportion of students having mastered skills does not change remarkably after 10 practices in the ASSISTments system and after 15 practices in the Cognitive Algebra system. And eventually there are approximately 25% of students in the Cognitive Algebra system and 35% of students in the ASSISTments system still at un-mastered state.

1.4. Adaptive Learning

Which tutorial sequences are proper for students? Several works were conducted recently to locate the activities where the learner is making high learning progress (Gottlieb, Oudeyer, Lopes, & Baranes, 2013; Lopes & Oudeyer, 2012; Oudeyer, Kaplan, & Hafner, 2007), based on the concept of Intrinsically motivating activities (Berlyne, 1960; Csikszentmihalyi & Csikszentmihalyi, 1992). They argue that students will learn at high efficient level when they are assigned with activities that are neither too easy nor too hard, but just above students' current ability, this type of activities is described as zone of proximal development (Lee, 2005). Based on this character, Lopes et al. propose an adaptive method, called "Right Activity at the Right Time" (RiARiT), in order to provide students with optimal learning items at each time (Lopes, Clement, Roy, & Oudeyer, 2013). Therefore, to assign appropriate sequences to students, we should follow two steps: student knowledge detection and tutorial adaption.

Student knowledge detection step is to estimate the values of variables that characterize students, like performance, knowledge level, score or mark. Detecting and predicting student knowledge is one of the most popular tasks in educational data mining, and numerous of different models and methods have been applied. Like Bayesian networks in (Baker, Corbett, & Aleven, 2008; Jonsson, Johns et al., 2005), logistic regression in (Beck & Gong, 2013; Feng & Beck, 2009; Wan & Beck,

2015), and sequential pattern mining in (Andrejko, Barla, Bieliková, & Tvarozek, 2007; Antunes, 2008).

The tutorial adaption step is to provide students with the tutorial sequences with respect to their current knowledge level detected in the previous step. The choices can be made in the sequences include: problems hardness, types of feedback, links to visit, and so on. Different data mining techniques have been used for this task. For example, Markellou et al. use association rule mining to produce recommendations for learning materials in e-learning system(Markellou, Mousourouli, Spiros, & Tsakalidis, 2005); neural networks and decision trees are used to provide personalized learning support in (Guo & Zhang, 2009); sequential pattern mining has been developed to customize learning content based on learning style and web usage habits (Ting, Ouyang, & Zhu, 2008).

In the work (Lopes, Clement, Roy, & Oudeyer, 2013), Lopes et al. use k-armed bandit algorithm to generate the next problem according to student's competences which is updated after every response. However, they estimate the competences without any features that describe students, but only problem hardness, which overlooks the fact that students are different in gender, prior knowledge, or other factors, and thus they need different tutorial style. For example, students with less preparation might need more explanatory learning content in the learning process.

In order to incorporate the students' characters into k-armed bandits, we could use contextual k-armed bandit, where the action is selected based on the contextual information about the actions or users. Want et al. investigate the effect of an observed variable that provides some information on the rewards to be obtained in the k-armed bandits problems (Wang, Kulkarni, & Poor, 2005). They find that the additional side information could significantly improve sequential decisions in bandit problems. Li et al. also utilize contextual bandits to recommend personalized news articles for each user in the work (Li, Chu, Langford, & Schapire, 2010). Nevertheless, rare work has been done to apply contextual k-armed bandits in ITSs.

1.5. Research Questions

This dissertation work focused on using contextual k-armed bandits to personalize tutorial strategy for each student in an ITS, in order to avoid over-tutoring or under-tutoring. Most of my work was and will be conducted on the ASSISTments. The main research questions investigated in this work include:

1. What context should be incorporated into the bandits?

There are different ways to describe students, are all these useful in contextual bandits? It is possible that the feature that enhances student models is useless in contextual bandits, because they are two different issues. For example, prior knowledge is an important feature in student model, but students with different level of prior knowledge might have the same treatment, therefore, it is not a good feature in contextual bandits. Thus, we need to carefully select the features for the bandits. More precisely, we should find a mechanism to evaluate the effect of a feature in personalizing tutoring strategy, before integrating it with bandits. In this work, I also need to consider if a context is too specific to make it meaningless. Moreover, I want to explore whether a bandit algorithm with the same context is effective in different data sets.

Another issue to be considered is the number of features. Too many features could cause overfitting in the bandit problems. Moreover, it requires large records to cover every possibility, and thus to make reasonable decisions. To overcome the drawbacks, we should limit the number of features incorporated into the bandits. Or we would make some assumptions, like in Naïve Bayes where features are conditional independent.

2. How shall we organize the context?

   After answering the first research question, it is very clear whether an individual feature is useful in the given data set. Another consequent problem to tackle is how to integrate multiple features with bandits. A simple solution is to separate the data into groups according to every possible combination of features values, and apply a k-armed bandit algorithm on each group. However, its complexity increases exponentially with respect to the number of available features.

   Another solution is to use traditional classification model to combine features with bandits, like LinUCB (Li, Chu, Langford, & Schapire, 2010) used linear regression combine features with UCB. In this work, I plan to use decision tree to handle multiple features. A challenge here is that the traditional decision tree algorithm is used in classification, and a trained decision tree has the least classification error in a given data. While in this problem, we focus on constructing decision trees that can discriminate students who have different treatments. Another challenge is how to update decision tree structure in the real sequential choice experiment, since more and more students' records are obtained, and we need to maximize the overall students' reward.

3. How much benefit?

Since it is time consuming to apply the contextual bandits on the real experiments to do the controlled comparison, in this work, we just do the simulation on the data from randomized control experiments in the ASSISTments. By compared with the original data set, we want to examine if using contextual bandits improves students' outcomes, and in what circumstances the improvements are made.

In the next chapter, I will introduce the background of adaptive systems. And then I will talk about student models that are used to detect students' status. In Chapter 4, I will discuss why we need k-armed bandits in ASSISTments, and answer the three research questions in details in the Chapter 5. Finally, I will point out possible directions of future works in Chapter 6.

CHAPTER 2

Background: Adaptive Systems

2.1. Introduction

Adaptive systems are referred to those systems attempt to be different for different students or groups of students, by considering information accumulated in the individual or group student models. From the perspective in the work (Brusilovsky, 1998), two key components in an adaptive system are user model and adaption based on the model. In this chapter, I will discuss the problem of "adaption", and user model in the next chapter.

Adaptive techniques can be useful to better support tutoring in ITSs for two reasons. First, the knowledge of different students can vary greatly. The same feedback can be unclear for a novice and at the same time trivial for an advanced learner. Second, the knowledge of a particular student can be different at different time. If a student is provided with problems at the same level, he would learn quite fast at the beginning, but then would find it uninteresting (Lopes, Clement, Roy, & Oudeyer, 2013) and show some off-task behaviors (Baker, 2007).

Researchers have been working on applying adaptive techniques in tutoring system for decades. Adaptive Hypermedia systems build a model of the goals, preferences and knowledge of the individual user and use this throughout the interaction for adaption to the needs of that user (Brusilovsky, 1998). In the context of educational hypermedia, the topics suggested to the learner for subsequent study would be determined by the learner's existing knowledge (Brusilovsky, 1998). AH aim at overcoming these problems by providing adaptive navigation support and adaptive navigation support and adaptive content (Kaplan, Fenwick, & Chen, 1993). The strategy termed as aptitude-treatment interaction (ATI) proposes different types of instructions or even different media types for different students (Burgos, Tattersall, & Koper, 2006). Gaze-reactive is used to evaluate student's aptitude in order to promote engagement and learning by providing dialog move that direct the student to reorient his/her attention (D'Mello, Olney, Williams, & Hays, 2012). Muldner et al. investigated the interactions between the interventions in the tutoring systems and students' affection (Muldner, Wixon et al., 2015).

Moreover, reusability of the adaptive learning models has been highly valued in many works. Koper pointed out that a designed learning notation "must make it possible to identify, isolate, decontextualize and exchange useful parts of a learning design so as to stimulate their reuse in other

contexts" (Koper, 2005). Aroyo et al. proposed a method that combined standard adaptive hypermedia model with semantic interoperability, which enabled the model to be executable with different approaches.

To analyze the adaption in ITSs, we should consider following questions: adapting to what? What can be adapted? What methods we should use? I will briefly explore answers to each question, with respect to the ASSISTments system, and talk about several existing methods in the following parts. Finally, I will give an example of selecting YouTube videos based on which skills students are learning.

2.2. Adapting to What?

This question can be expressed in another form: what aspects of students shall we consider in the adaptive tutoring systems? Which features can be used to differentiate students? As categorized in the work (Brusilovsky, 1998), five common features are considered in the adaptive hypermedia systems: user's goals, knowledge, background, hyperspace experience, and preferences. In this dissertation work, we just analyze and use three features that are most related to the ASSISTments system: student's goals, knowledge, and background.

2.2.1. Goals

Student's goals or tasks are a feature related with the context of a student's work in ITSs, rather than with the student as an individual. In some systems, it is reasonable to distinguish the level of goals, where low level goals can change quite often and high level goals are more stable. For example, in the tutoring systems, learning goal is always a high-level goal, while the problem-solving goal is a low-level goal which changes from one problem to another.

According to which aspects we focus on, student's goals in ITSs could be different, such as mastery speed in (Botelho, Wan, & Heffernan, 2015), performance in posttest or retention test (Li, 2013; Xiong, Wang, & Beck, 2015), and engagement in (Baker, 2007; D'Mello, Olney, Williams, & Hays, 2012). In this dissertation work, I mainly consider student's performance in the posttest and mastery speed in the learning process as student's goals, and analyze how to adapt tutorial strategies with respect to different goals.

2.2.2. Knowledge

Student's knowledge appears to be the most important feature of the student in the adaptive tutoring systems. There are various student models have been developed in past works, which recognize changes of student's knowledge and update the model accordingly. The student's knowledge is

represented as different concept in the models, like wheel spinning status, skill mastery level, or affections. Those concepts can be binary value (known and unknown), a qualitative value (good, average, and poor), or a quantitative measure (a probability that a student will be wheel spinning on a skill).

The student's knowledge can also be expressed as a function of a set of student-related features. For example, the probability of wheel spinning is a logistic regression function of several features, 9 features in (Beck & Gong, 2013) and 10 features in (Wan & Beck, 2015). A challenge of measure student's knowledge in an experiment is to determine which student features are related and proper to be used. Fewer feature might not be enough to capture the changes of student's knowledge, such as adding prerequisite factor or general learning ability into the wheel spinning model improves the model accuracy (Wan & Beck, 2015); on the other hand, more features requires much more student records, otherwise, overfitting would occur. In the Chapter 5 I will discuss which kinds of features will be considered to express student's knowledge in details.

2.2.3. Background

Student's background is referred to the information related to the student's previous experience outside the subject of the tutoring systems. A common used student's background in the ASSISTments is the class feature. Since the students in the same class are taught by the same teacher and normally get the same instructions, they might share some patterns in the learning process. Therefore, to predict a student's outcome, his peers' performance could give a clue.

Wang and Beck incorporated the parameters at the class level into the Bayesian knowledge tracing models (Wang & Beck, 2013), by compared with the parameters at student level and skill level, they found a plausible result that the prior knowledge parameter (k_0) derived from class information makes better models, this means that the students in the same class have similar initial knowledge. Xiong et al. investigated the class effect in the retention model (Xiong, Beck, & Li, 2013), their result, adding the class features slightly improves the retention model, suggested that there seemed to be an overall class effect that differs from average performance on other skills.

2.3. What Can be Adapted?

An important issue should be considered in any adaptive systems is: what features of the systems can differ for different students? What different tutorial sequences can the systems offer? In this dissertation, we mainly focus on assigning students into which conditional problem sets, and the problem sets differ in feedback type or problem content.

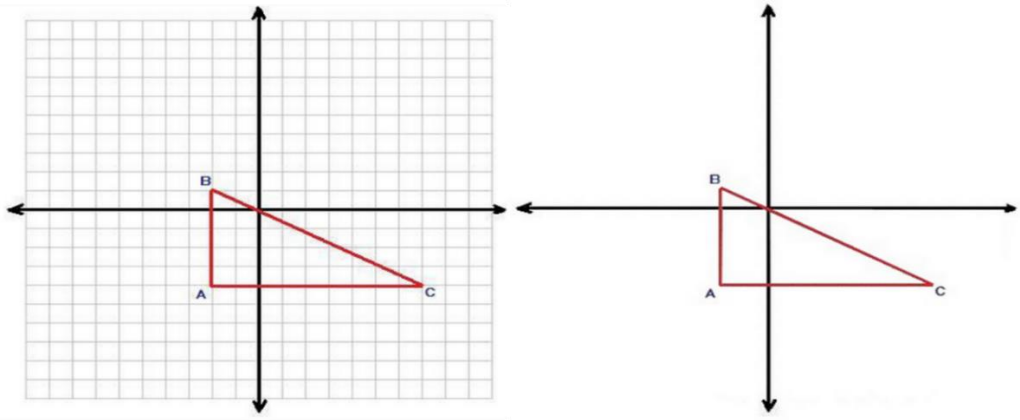


Figure 2. An example of a problem with two versions in the ASSISTments.

Problem content is the way to describe the problem, including problem structure (fill in blank or multiple choices or other type), the format of numbers in the problem (such as fraction, percent, or decimal), and other information like videos, figures, or links. A student's performance may vary greatly in different versions of problem. For example, considering the two types of problem in the Figure 2, it is better to provide the student with low spatial sensitivity with the problem with grids, e.g. the left problem.

As introduced in Chapter 1, there are three main types of feedbacks in ASSISTments: scaffolding problems, hints, and extra tutorial material. Feedbacks provide extra explanation about the problem or the corresponding skills, which should be adapted according to student's current knowledge, goals, and other characteristics of the student. For example, a qualified student can be provided with deeper information while a novice needs additional explanation. Ostrow and Heffernan propose a study of a randomized controlled trial that is used to validate the effect of feedback type and the effect of student choice within ASSISTments (Ostrow & Heffernan, 2014). Their result shows that students have significantly higher correctness if provided choices of feedback, and they prefer to choose video feedback.

2.4. What Method Shall We Use?

Like the student modeling, some other data mining techniques also aim at describing students' special characteristics or grouping students to customize needs for students, such as clustering, association rule mining, and sequential pattern mining in tutoring systems (Koutri, Avouris, & Daskalaki, 2005).

Clustering is an unsupervised classification technique that groups a set of unlabeled objects into clusters where the objects in the same cluster are more similar to each other than to those in the

other clusters. As stated in (Koutri, Avouris, & Daskalaki, 2005), clustering is useful in three different aspects of adaptions in web-based tutoring systems: personal recommendation, dynamic adjustment, and static page adjustment. For example, Mobasher et al. propose an effective method, which focuses on capturing commonalities in the usage models with the ultimate goal to perform adaptive navigation support (Mobasher, Cooley, & Srivastava, 1999). Tang et al. construct a clustering method based on the sequence and the content of pages users visited, in order to promote group-based collaborative learning and to provide incremental learner diagnosis (Tang, Lau et al., 2000).

Association rule mining refers to the identification of all associations among certain data items, always expressed as if-then statement, so that the appearance of one subset of items in a transaction implies the appearance of other corresponding items (Agrawal, Imieliński, & Swami, 1993). A number of techniques discover association rules from different types of student data. Romero et al. discover association rules from students' usage information by using Grammar-Based Genetic Programming with multi-objective optimization techniques, in order to provide feedbacks to courseware authors (Romero, Ventura, & De Bra, 2004). Freyberger et al. mine association rules from a dataset derived from student-tutor interaction logs, in order to guide the search of transfer models (which map the questions in an ITS and the necessary skills to answer a question correctly) (Freyberger, Heffernan, & Ruiz, 2004). Lu applies a fuzzy association rule mining technique in a web-based learning recommendation system, in order to supply suitable materials to best meet each student need (Lu, 2004).

Sequential pattern mining, can be considered as a more restricted from of association rule mining, is to find patterns in an ordered list of items, such as presence of a set of items in a particular order or with time stamp (Agrawal & Srikant, 1995). The extraction of sequential patterns has been used in tutoring systems for discovering and comparison with expected behavioral patterns specified by the instructor that describe an ideal learning sequence (Pahl & Donnellan, 2002). Other research works use sequential pattern mining to: discover group interaction sequences indicative of problem, in order to assist student teams in early recognition of problems (Kay, Maisonneuve, Yacef, & Zaïane, 2006); extract frequent learning patterns from sequential learning sequences to group learners with good learning performance into several meaning clusters (Wang, Weng, Su, & Tseng, 2004).

Another method which is gaining popularity in educational data mining and intelligent tutoring system is k-armed bandit algorithm. It is first used in a problem of slot machines in which a gambler

at a set of slot machines has to decide which machines to play and how to play at each time. When applying the algorithm in tutoring systems, the different tutorial strategies can be regarded as slot machines or actions one of which is chosen for a particular student at each time. Based on student's current competence, Lopes et al. use a k-armed bandit algorithm to select the components with appropriate difficulty level to generate a problem for the student, and then the student's response to that problem is used to update his/her competence level (Lopes, Clement, Roy, & Oudeyer, 2013). Students tutored by this strategy is proved to have better performance than by random generated problems in their work. Without any information about the problem, Clement et al. propose another algorithm, ZPDES, based on the zone of proximal development (Lee, 2005) and the empirical estimation of learning progress (Oudeyer & Kaplan, 2007), to choose exercise at each step (Clement, Oudeyer, Roy, & Lopes, 2014).

In this dissertation work, I will use k-armed bandit algorithm in the ASSISTments experiments and analyze the effect of different student-based features and skill-based features in the bandit. The bandit algorithm and the features will be discussed in the Chapter 4.

2.5. Example: YouTube Video Selection

2.5.1. Background

After many trials on learning a skill in a tutoring system, a student cannot still reach mastery, then this student is probably wheel spinning on the skill (Beck & Gong, 2013). Beck and Gong construct a wheel spinning model to predict whether a student would be wheel spinning on a skill at early learning stage, and they find a strong connection between "gaming" the system and wheel spinning (Beck & Gong, 2013). Wan and Beck discover that a student with lower performance on the prerequisite skills is more likely to be wheel spinning on the post skill (Wan & Beck, 2015). However, they do not provide any approach to prevent the wheel spinning cases. A plausible solution to this issue is to provide a student with proper tutorial feedbacks to "cure" the student when detecting he is likely to be wheel spinning.

Feedback plays an important role in tutoring systems that informs students about how they perform currently or provides students with helpful hints or tutorial content when they are struggling in learning. It does not only enhance students' learning results (Fossati, 2008; Narciss, 2013; Roscoe, Snow, & McNamara, 2013), but also influences students' affection in the learning process (Heylen, Vissers, op den Akker, & Nijholt, 2004; Robison, McQuiggan, & Lester, 2009).

According to the review of feedback in Kulhavy and Stock's work (Kulhavy & Stock, 1989), effective feedback contains two types of information: verification tells whether an answer is correct

or not; elaboration provides more illustrative information about the question or related topics to guide student to the correct answer. In the ASSISTments skill builder, of which experiment data we use in this work, students can know immediately if get a correct answer. Thus, to prevent wheel spinning, verification feedback is not enough, but feedback with more explanatory content is necessary for those students who are likely to wheel spinning, to increase their knowledge and understanding in the skills.

There are different types of elaboration feedback, like scaffolding questions, hints, and worked examples. Since hypermedia is being more and more popular, many researchers are working on applying tutorial hypermedia in the tutoring systems. Kelly et al. study a controlled experiment in which a student is provided with either a YouTube or a motivational message to persist with the learning session (Kelly, Heffernan, D'Mello, Namais, & Strain, 2013). Their results show that students with video feedback have higher homework completion rate. By compared with the blank text, Ostrow and Heffernan find that video feedback provides better understanding and thus enhances students' learning outcomes (Ostrow & Heffernan, 2014).

In ASSISTments, students are provided with several links to tutorial YouTube videos when they are struggling in learning. To analyze the tutorial effect of these YouTube videos, we will use the wheel-spinning models defined in (Wan & Beck, 2015). In this work, we mainly focus on the following questions:

1. Are the videos really helpful in curing wheel spinning? To answer this question, we will compare the students' performance before and after viewing the videos according to the wheel-spinning model.
2. What factors make a video with better tutorial effect? We will use text mining and web mining techniques to generate several features from the videos and related contents, and then investigate the relation between these features and the students' outcomes.

2.5.2. Method

The predicted value from wheel-spinning model indicates how likely a student would be wheel spinning on a skill. If a video really helps a student who had trouble in learning a skill before watching the video, then a shift downward of the probability would start at the time viewing the video. Therefore, we will use the trend of the probability estimated at each opportunity in the wheel-spinning model (Wan & Beck, 2015) to measure the tutorial effect of YouTube videos. To investigate what factors make a "better" video, the first step is to abstract video features. The following sections explain how we generate features from YouTube videos.

Resources

In this work, we generate three kinds of features: YouTube basic features, students related features, and text-based features. The first kind of features are collected directly from YouTube and the second kind of features are generated from students' activities. These features are:

- View count: how many times a video is viewed by YouTube users.
- Rate count: the number of YouTube video ratings.
- Rate score: the final rate score of the video.
- Video duration: how long the video is.
- Stay time: how long a student stay in the video web page.
- Viewed proportion: this is calculated as (stay time)/ (video duration).
- Student's opinion: after viewing the video, the ASSISTments provide a survey question asking students if the video is useful.

The last one is more complicate. We will use text mining techniques to abstract features from four YouTube web contents: video title, video description, closed captions, and YouTube user comments. Finding educationally effective webpages is a somewhat different problem. Presumably learners are interested in finding pages that are engaging, contain good explanations, and will leave the learner understanding the subject he set out to learn. These webpages should contain the information related to the topics students are learning. Since the data in this work is collected from students' responses in learning the mathematic skills in the ASSISTments, we will capture the relationship between YouTube webpage contents and a set of mathematic terms as features. These mathematic terms are taken from ("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,")("Mathematics terms,") which are separated into six subjects and cover almost all the topics in our data.

Pre-processing

1. Stemming

An English term can have different inflected words; the stemming is to identify those words derived from the same terms. For example, "so" and "sooo" probably have no difference. In this work, we will use the snowball stemmer from the NLTK package (Bird, Klein, & Loper, 2009) to transform the words in the video-related contents into their root terms. This pre-processing step would reduce mismatches in the process of counting occurrences of mathematic words and skill names in the video contents.

2. Grouping mathematic terms

This step is to assign each skill in our data set into one of the six subjects in the mathematic terms, the terms in the assigned subject is called related terms and others unrelated terms for the skill. The objective of this step is to provide a way to validate if the video is talking about the skill a student is acquiring or general mathematic topics.

Generated text-based features

Since for each video visiting record, the visited video relates to a skill a student is learning. To abstract the text-based features as the relationship between skills and videos, we will match the skill-based text with the video-based text. The skill-based text contains skill name, skill-related mathematic words, and skill-unrelated mathematic words. The video-based text contains video title, video description, video closed-captions, and YouTube users' comments. The matches are used to generate the following four types of features:

1. #Type: it is number of different mathematic terms or skill names occur in the content. If more than half of words in a math term or a skill name exist in the content, then we say this math term or skill name is in the content. For example, the math term "fraction division" is in the string "one fraction is 1/3". Obviously, the value of this feature for the skill name is 1 or 0, because each skill has only one name.
2. #Token: it is number of words in the video content match the skill-based text. In the previous example, the value of this feature in this example is 1, because there is only one word in the string, "fraction", appears in the math term.
3. Proportion of skill-based text: this is calculated as, #type/ (total terms in the skill-based text). If there are five skill-related math terms and only one is matched in the video description, then the value of this feature is 0.2.
4. Proportion of covered video-based text: this is calculated as, #token/ (total words in the video-based text). In the previous example, match the math term "fraction division" to the string "one fraction is 1/3". There are four words in the string, and only one word appears in the math term, so the value of this feature is 0.25.

Since there are three different types of skill-based text, four types of video-based text, and four types of matching features, there are 36 (3*3*4) features generated in this step.

2.5.3. Results

Data

The dataset used in this work in collected from students' activities in learning math skills in ASSISTments in 2013. This dataset if from students selecting to see a YouTube video with 2-5 webpages human-selected for each skill in ASSISTments. Since the web-help functionality was available in ASSISTments for a short term, our data contains only 628 YouTube video related student-skill pairs.

Before analyzing the video tutoring effect, we should notice that our sample of viewed webpages is not random sampling of students. In our intuition, the tutorial YouTube videos are viewed by those students who have trouble in and are also interested in learning the domain knowledge, rather than getting through the assignments. To validate this, we compare students' performance on four different student sets:

- Set 1: all students in ASSISTments in 2013.
- Set 2: those students who are wheel spinning on at least one skill.
- Set 3: those students who have view a YouTube video at least once.
- Set 4: those student-skill pairs such that the student visit a YouTube video in the process of studying the skill.

The result, shown in Table 1, contains three different student-skill pairs, wheel spinning pair, master pair, and indeterminate pair. In a wheel spinning student-skill pair, the student is wheel spinning on the skill. Respectively, the student masters the skill in the master pair. In an indeterminate student-skill pair, the student does not master the skill and he drops out before the 10th opportunity. This case is considered as wheel spinning in the work (Wan & Beck, 2015).

As shown in the table, the wheel spinning rate in the set 4 is much higher than in the other student sets, which supports our guessing – the videos are viewed by those students who are struggling. Another interesting finding that indeterminate rates in the set 3 and set 4, in which students once visit a video in while learning a skill, are lower than in the other two sets. This indicates that these students are more willing to learn the knowledge, so they are less likely to drop out learning skills.

Table 1. Distribution of student-skill pairs in each bin. An indeterminate student-skill pair mean the student does not master the skill and he drops out before the 10th opportunity. For example, in the first row, 7.5% of student-skill pairs in the whole dataset is wheel spinning case, 69.4% is mastery case, and 23.1% is indeterminate case.

	Wheel spinning pairs	Master pairs	Indeterminate pairs
Set 1	7.5%	69.4%	23.1%
Set 2	10.4%	67.5%	22.1%
Set 3	11.0%	75.4%	13.6%
Set 4	30.1%	56.4%	13.5%

Experiment 1: tutorial efficacy of videos

As aforementioned, when a student views a video, he is probably struggling, and he is also willing to learn the knowledge. To check if the video helps the student, we need to estimate the student's outcome if he did not view the video. A proper way to do this is to select other students who have the similar situation before visiting the video but choose to keep practicing without visiting the videos, and then use their outcomes as the estimated value.

In this step, we first use the wheel spinning model to make prediction for each student's practice, and then for each student who visit a video when learning a skill, we select such students whose predicted wheel-spinning value is similar with the former student's value at the opportunity he views the video (difference is no more than 0.01). For example, a student S1 visits a video at the 3rd opportunity, and his predicted wheel-spinning value at that time is 0.3. Then the selected students' predicted values at the 3rd opportunity are in the range 0.29-0.31, and they do not visit the video in the learning process. Finally, we use the wheel spinning ratio among those selected students as the estimated outcome for the former student if he did not visit the video.

After filtering the similar students, we then compute the wheel spinning ratio for the video-helped students and the similar students. Here we consider the indeterminate cases as wheel spinning like the work in (Wan & Beck, 2015). The wheel spinning ratio for the video-helped students is 0.436, and 0.428 for the similar students, which shows no much difference between them. One possible explanation for this is some students ask for help too late. If a student realizes he need extra video for help after many trails in learning a skill, then he has very high probability to wheel spin on the skill since we regard the students who do not achieve mastery level in 10 opportunities as wheel spinning.

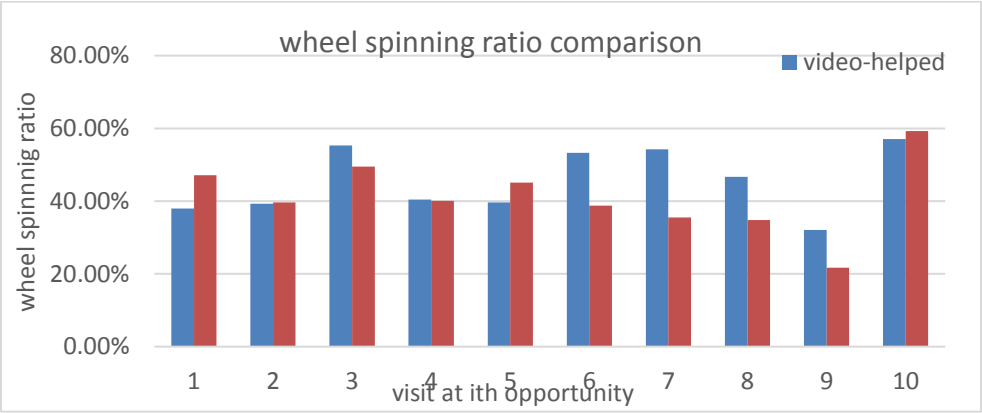


Figure 3. Wheel spinning ratio comparison between the video-helped students and similar students.

From the result in Figure 3, the explanation seems to be reasonable, because the wheel spinning ratio for the video-helped students is higher than the similar students if the video is viewed at 6-9th opportunity. Therefore, this arises another question: does visiting video at the early stage help students in learning? When visiting videos at the 1st opportunity, students will get about 10% less wheel spinning ratio. However, visiting the tutorial videos at the other times does not implies remarkable improvements on wheel spinning. Thus, there might be other factors influencing videos tutoring efficacy.

Experiment 2: feature selection

To answer the second question: what factors make a video with better tutorial effect? Here we still use the predicted value from the wheel spinning model. If a student has learned knowledge from a video, then his predicted wheel spinning value should decrease after viewing. Therefore, we define a target class, a video is helpful for a student or not, as if the predicted wheel spinning value at the last opportunity is less than at the opportunity the student views the video. For example, if a student views a video at 3rd opportunity, and the predicted value from wheel spinning model at that time is 0.4, and 0.2 at the last opportunity, then we say this video is helpful for the student. To simplify our problem, we just disregard the cases students visit videos at the last opportunity in this step.

After constructing target class for each student-skill pair, we then apply feature selection algorithms over the 43 features as introduced in the Method Section. In this work, we use three feature selection algorithms from WEKA (Hall, Frank et al., 2009): CFS + Best First, CFS + Rank Search, and Ranker + Information Gain. The first two algorithms will output a set of features that is most related with target class, while the last one will rank the features according to the evaluation function, and we will keep the top 10 features in the final feature set for this algorithm. With 10-fold cross validation, there are 3 features that appear in the selected feature sets of all three algorithms:

- The proportion of skill-unrelated math terms that are covered in the video comments (the 3rd type of feature with matching skill-unrelated math terms with video comments).
- The number of words in the video comments that match the skill-related math terms (the 2nd type of feature with matching skill-related math terms with video comments).
- The number of words in the video description that match the skill-related math terms (the 2nd type of feature with matching skill-unrelated math terms with video description).

Experiment 3: effect in the wheel-spinning model

In the previous experiment, we evaluate the relation between video based features and the wheel spinning improvement. In this experiment, we will add the related features into the wheel spinning

model, to validate the effect of such features in the wheel-spinning model. If a feature plays an important role in evaluating impact the videos are having on student learning, then it improves the wheel-spinning model.

Since less than 1% of student-skill pairs in the whole data are related with YouTube video visiting, the effect of web based features would be tiny over the whole data set and the models measurement would be almost the same. Thus, we construct models over only the YouTube web visiting related student-skill pairs. We construct three different wheel-spinning models: model without video-based features, model with all video-based features (43 features generated by our method), and model with the selected features (as state in the previous experiment).

The results in Table 2 indicate that adding video-based features improves wheel-spinning model accuracy, which means these features are related with wheel spinning, and we could use them to evaluate video tutoring effect. However, from the observation that measurements of the 2nd and the 3rd model are very close, we can get that the most impact attributed to tutoring efficacy is how many proportions of math terms and how frequently they are covered in the video-based contents. Moreover, the coefficients of the three features in the model are all negative, which means a better video should cover more math terms in the video-related texts.

Table 2. Measurement of three different wheel-spinning models.

	Cox and Snell R^2
Model without video-based features	0.354
Model with all video-based features	0.364
Model with the three-selected video-based features	0.362

2.5.4. Discussion

Being increasing popular, obtaining tutoring contents from internet through electronic devices such as personal computers, smartphones, and other portable devices, is an important way for students to learn knowledge, and these web-based tutoring contents can be necessary complementary of school tutoring. The work in (He, Swenson, & Lents, 2012) proves that incorporation of video tutorials as a supplement to learning in undergraduate analytical chemistry course increases students' comprehension in difficult concepts. ASSISTments also provides videos as extra tutoring material in the learning process. In this work, we analyze how students use the videos and the effect of videos in tutoring students.

Our data indicates that students view the videos when they are struggling and they hope to learn the knowledge through the videos. However, about only half of students eventually master the skills. A further analysis suggests that students should ask for the videos to help them at the early stage. However, students might not realize they are unable to master the skills very fast. Thus, it requires our tutoring systems to detect their wheel-spinning status as sooner as possible and then to provide appropriate extra assistances.

With using feature selection algorithms, the selected video-based features are all related with the math terms. By incorporating them into wheel-spinning models, model results imply that a video with stronger connection with the math term is more likely to heal students' wheel spinning. In summary, our work provides a guideline of how to evaluate the video effect in tutoring students, and can be useful in selecting the proper videos for students in the future.

However, there are still challenges in future works. The first is lack of data. Current webpage visiting data contains 5638 students' records, this is very rare comparing to the whole data with over 2 million records, and 628 student-skill pairs compare to 340 thousand pairs. Thus, models constructed on this sparse data is unstable in generalization, and is not reliable in evaluating webpage efficacy in the future. Moreover, the webpage visiting data contains 52 different skills and 58 skill-webpage pairs, which means most of skills are associated with only one YouTube page, and it prevents us from analyzing if different webpages have different tutoring efficacies for a specific skill.

The second is how to construct a model to directly evaluate videos tutoring effect. From the results, we can see that some web based features have high coefficients, which means they are important in WS models. However, there is no clear evidence that these features are also useful in evaluating webpage tutoring efficacy. In additional, if we want to compare different webpages' effect, other questions arise: If we use the web based features to construct webpage evaluation model, what does the model look like? Is the feature with higher coefficient in WS model more important in the evaluation model?

CHAPTER 3

Student Modeling

3.1. Introduction

Student models generally represent inferences about users, relevant characteristics of users, and users' records, particularly past interactions with system. The main objective of student modeling is to develop cognitive models of human users or students, including a modeling of content skills, knowledge about learning, and affective characteristics. Various algorithms have been applied to consider different students' characteristics (like learning styles, affections, and motivations) and other factors, to automatically model their knowledge. In this section, I will discuss several student models and the factors used in the models.

3.1.1. Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) is an approach that relies on Bayesian theory. BKT uses two states to in the model: learned state and unlearned state, which indicates whether a student has mastered a skill or not. And the student's performance is made based on which state he is currently in, as illustrated in Figure 4. BKT assumes that students can transit from unlearned state to learned state through practicing, but cannot in the opposite way. Their transition and emission diagram is shown in Figure 5. The four parameters in Figure 5 are:

- $P\ (L_0)$: Probability of the skill is already mastered before the first practice of the skill;
- $P\ (T)$: Probability of the skill will be learned at each practice;
- $P\ (g)$: Probability of the student will guess correctly if the skill is not mastered;
- $P\ (s)$: Probability of the student will fail even if the skill is mastered, called slip.

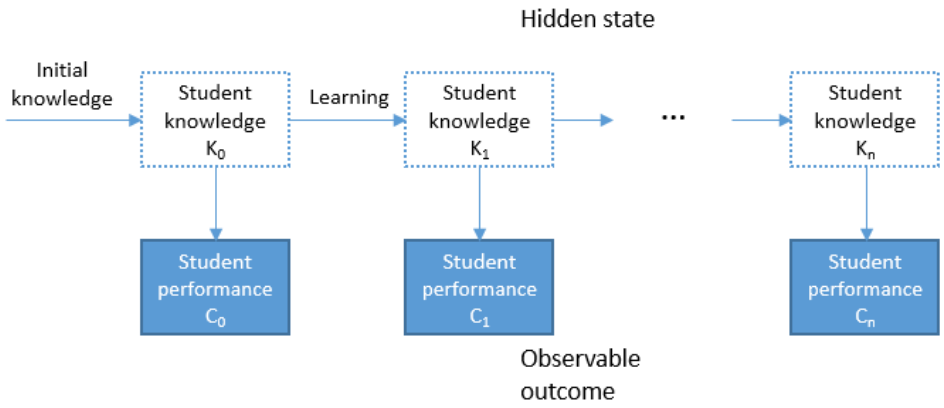


Figure 4. Illustration of Bayesian Knowledge Tracing.

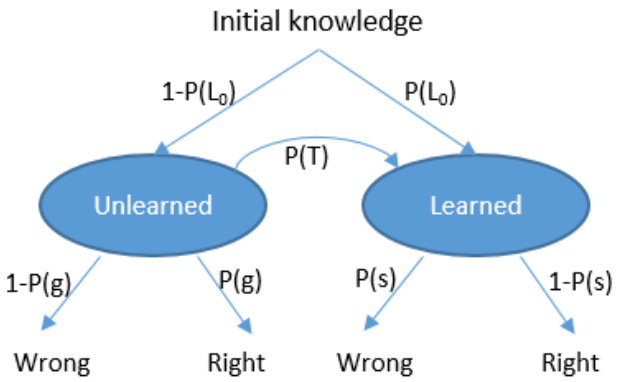


Figure 5. BKT transition and emission diagram.

The key factor in this model is the student's performance – right or wrong in each question. The main benefit of the BKT model is to monitor changes in student knowledge state during practices. However, since the BKY model is always trained on a set of students' practices on one skill, it lacks the ability to handle multi-skills simultaneously and the ability to regulate students' personalities. Therefore, some variants of BKT use different factors, such as student-specific parameters (Lee & Brunskill, 2012; Pardos & Heffernan, 2010; Yudelson, Koedinger, & Gordon, 2013), class feature (Wang & Beck, 2013), and partial credit (Wang & Heffernan, 2013).

Advantages and Disadvantages

The main benefit of the BKT model is that it monitors changes in student knowledge state during practice. Each time a student answers a question, the model updates its estimate of whether the student knows the skill based on the student's answer. Other works with BKT introduce a number of advances, for example, estimates of guessing and slipping in contextual models (Baker, Corbet, & Aleven, 2008), estimates of probability of transition from use of help features (Beck, Chang, Mostow, & Corbett, 2008), and estimates of the initial probability that the student knows the skill (Pardos & Heffernan, 2010).

BKT suffers two major problems with parameter estimates (Beck & Chang, 2007): local maxima and multiple global maxima. The first one could be solved by multiple restart. The second one refers to different sets of parameters that could fit the same data equally well. The other major problem of BKT is its lack of ability to handle multi-skills simultaneously, because BKT works by looking at historical observations on a skill.

3.1.2. LFA & PFA

Learning Factor Analysis model is used to modeling students' knowledge in multiple skills with a Q-matrix (representation of the KC, knowledge component, to item assignment) (Cen, Koedinger, & Junker, 2006). This model captures three important factors influencing the learning and

performance of KCs, as depicted in the following form, α_i – each student's preparation, β_j – hardness of each KC, and $n_{i,j}$ – each student's learning rate on each KC. The form is:

$$m(i, j \in KCs, n) = \alpha_i + \sum_{j \in KCs} (\beta_j + \gamma_j n_{i,j})$$

$$p(m) = \frac{1}{1 + e^{-m}}$$

PFA is considered as an extension of LFA, which was presented by Pavlik et al. (Pavlik, Cen, & Koedinger, 2009). Different from LFA, the parameter representing student's ability is removed in PFA. The other difference is that instead of using only one parameter to capture student's prior practice, the author uses two parameters to track student's prior successes and prior failures. Because correct responses may lead to more learning than incorrect responses. The PFA form is:

$$m(i, j \in KCs, s, f) = \sum_{j \in KCs} (\beta_j + \gamma_j s_{i,j} + \rho_j f_{i,j})$$

$$p(m) = \frac{1}{1 + e^{-m}}$$

In the equation, m is a logit value representing the accumulated learning for student i using one or more knowledge components (KCs) j. The easiness of these KCs is captured by the β parameters for each KC, s tracks the prior successes for each KC, and f tracks the prior failures, and γ and ρ scale the effect of these observation counts.

Advantages and Disadvantages

As discussed in previous section, BKT suffers the problem of multi-skill performances. PFA and LFA) can be used to tackle this problem. As we can see in the PFA formula, the skill effects of and student's performances on all related skills are combined to predict student's outcome for current item. Another benefit of PFA is that it produces a real valued estimate of strength for each skill.

One drawback of LFA (or PFA) is it would produce negative learning rate due to overfitting. The solution of this is to set 0 as lower bound. This model counts the numbers of prior successful and failed practices; however, it doesn't consider the order in which these practices occurred. Suppose we have two students working on the same question set. The outcome of first student is: incorrect, incorrect, correct, correct, and correct. While the outcome of second student is: correct, incorrect, correct, incorrect, and correct. In our intuition, the first student should have higher mastery level than the second student. But in PFA, they are counted as the same.

3.1.3. Wheel Spinning Model

In ITSs, wheel spinning refers to that a student cannot master a skill no matter how many practices he tries. In the real experiments, it is inappropriate and impossible to force students to keep practicing until they achieve mastery. Therefore, we need to quantify the practices that if a student cannot achieve mastery in the number of practices, then he is wheel spinning on that skill.

Beck and Gong found that if a student cannot master a skill in 10 questions, then he has very low probability to get mastery later (Beck & Gong, 2013). So they used 10 questions as the threshold. They also found that about 25% of students in the Cognitive Algebra system and 35% of students in the ASSISTments system are wheel spinning on the corresponding skills. And they constructed a wheel spinning model, a logistic regression model, to predict the wheel spinning cases based on following factors:

a) The number of prior correct responses by the student on this skill. This feature is proved useful in the Performance Factors Analysis model (Pavlik, Cen, & Koedinger, 2009).
b) The number of problems in a row correctly responded by the on the skill prior to the current problem. In the ASSISTments, mastery is defined as 3 correct responses in a row. Thus, the number of consecutive correct responses is an important factor. The value of this feature is from 0 to 2.
c) The exponential mean Z-score of response times on this skill. The response time for each item is transferred into a Z-score, and then exponential mean is calculated for each student by: $\gamma * \text{prior\_average} + (1-\gamma) * \text{new\_observation}$, with $\gamma = 0.7$.
d) The exponential mean count of rapid guessing. This measures how often the student was rapidly guessing.
e) The exponential mean count of rapid response. This measures how often the student took a rapid response. This feature as well as the feature (d) reflects how serious the student is learning the skill through the tutoring system. Similar features related with "gaming" the system were used in gaming detectors as in (Arroyo & Woolf, 2005; Baker, Corbett, Roll, & Koedinger, 2008; Gong, Beck, Heffernan, & Forbes-Summers, 2010)
f) Count of bottom-out hint. The number of times the student reached a bottom-out hint on this skill prior to the current problem.
g) The exponential mean count of 3 consecutive bottom-out-hints. This measures how often the student reached bottom out hints on 3 consecutive problems.
h) Skill identification.
i) Prior response count.

In additional to wheel spinning prediction, they found a positive correlation between wheel spinning and gaming – students with higher gaming score are more likely to exhibit wheel spinning. This relationship is also hold for a particular student that a student is gaming more in the skills he would wheel spin than in the skills he would master.

BKT, LFA and PFA are used to model students' knowledge level in skills, but they failed to determine if a student needs help when he/she is struggling in learning a skill. The main benefit of wheel spinning model is to detect if a student needs help in the early learning stage, therefore, proper interventions can be provided to the students.

3.1.4. Big Drawbacks

The models talked in the previous section are all based on the factors in the learning progress of current skills. These models can be used to predict how well or how bad a student is learning skills. They can also be used in estimating how likely a student could guess and slip (Baker, Corbett, & Aleven, 2008) and how student's actions (successes or failures) affect the mastery of skills (Pavlik, Cen, & Koedinger, 2009). However, these models have two main drawbacks. First, they overlook the effect of skill relationships. Second, they don't know what else to do if students are sick.

1) Overlook the effect of skill relationships

   In pedagogical design, students usually learn skills in sequence since preliminary skills need to be learned prior to the complex skills. For example, students should know how to do squaring and square root before learning Pythagorean Theorem, as in Figure 6. The prerequisite-post relations underlie the design of learning sequences and adaption strategies in ITSs, and give an insight to estimate the knowledge in current skills from the prerequisite skills.

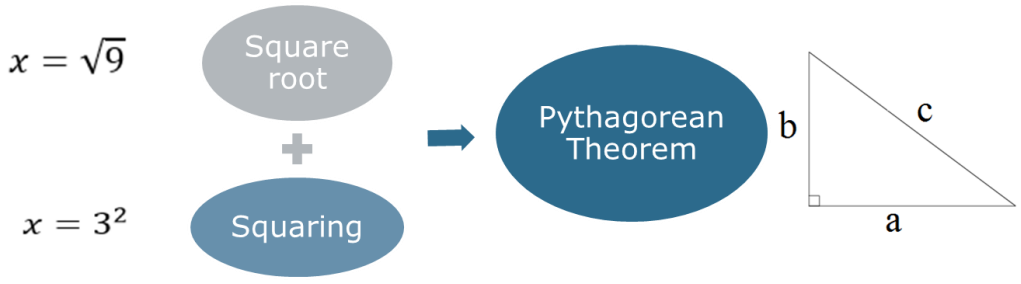


Figure 6. An example of prerequisite structure.

Many educational data mining techniques have been applied to discover and refine the prerequisite structures. The Partial Order Knowledge Structures (POKS) learning algorithm is used to learn the prerequisite structure at the item level in the work (Desmarais, Meshkinfam, & Gagnon, 2006); dynamic Bayesian networks are used to describe the hierarchy and relationships between different skills in (Käser, Klingler, Schwing, & Gross, 2014); Chen et al. build prerequisite relations between skills from the probabilistic knowledge states of students with association rule mining (Chen, Wuillemin, & Labat).

2) Don't know what else to do if students are sick

These models provide information about students' knowledge status at each step, but nothing about what to do if students are struggling in learning skills. To improve this, Rollinson and Brunskill propose a policy that suggests to stop tutoring students in ITSs when the estimations of students' knowledge from the models do not changes any more (Rollinson & Brunskill, 2015).

To overcome the first drawback, the skill-relation structure can be combined into the models. In previous works, I have used prerequisite performance to predict students' initial knowledge in the post skills, and I will talk two models in the next sections.

In this dissertation work, I will focus on providing students with the appropriate tutorial strategies according to their current knowledge status, especially when they are in trouble, and I will talk in detail in Chapter 4 and Chapter 5. To be clear, I do not generate or create the strategies (like hints, feedbacks, and problems) in this dissertation work, but select the optimal one for each student from the options.

3.2. Prerequisite Effect in Predicting Initial Knowledge

3.2.1. Introduction

Since in the KT model, initial knowledge is represented by a parameter $P\ (L_0)$, the probability of mastering the skill (Corbett & Anderson, 1994). As such, KT is often used to estimate each student's initial knowledge (Pardos & Heffernan, 2010). In the standard KT model, the parameter $P\ (L_0)$ is trained on all students' records in a training set, and assumes that all students have the same initial state of knowledge. However, this assumption is too strong to use the model to predict each individual student's first response. To overcome this drawback, Pardos and Heffernan use three heuristic functions to model individualization in KT (Pardos & Heffernan, 2010), and find that the method, setting initial individualized knowledge based on individual students' performance over all skills, yields superior results. This approach, however, overestimates the relationships

between skills. If learning a skill does not promote, or even hinder (Cree & Macaulay, 2002), learning another skill, then it is not appropriate to use knowledge in one skill to estimate another.

In the work (Botelho, Wan, & Heffernan, 2015), we utilize prerequisite structure, defined by domain experts, to predict student initial knowledge on subsequent skills, and we compare our method with the Knowledge Tracing (KT) model. Due to human effect, some skill relationships might be overestimated, or they may not exist in other applications. As such, we are seeking to answer the following two questions in this paper: 1. Does prerequisite information really help to improve the estimation of initial knowledge on subsequent skills? 2. Are all prerequisite relationships reliable?

3.2.2. Methodology

The method of prediction here utilizes a categorization of students based on the number of attempts taken to master each prerequisite skill. This type of methodology, often referred to as binning, as it places students into a set of finite bins or categories for which a prediction or inference could be made. Using information drawn from prerequisite skills, a prediction table can be constructed to model the initial knowledge of different types of students defined by separate bins.

As a measure of performance, we chose to use the number of attempts, or responses, made before mastering a prerequisite skill to categorize the students in our dataset; this number of attempts is often also referred to as mastery speed. This concept of a mastery status is common in intelligent tutor systems and is, in the case of ASSISTments, gained through three consecutive correct answers. Using the number of attempts to gain mastery status, our method calculates the overall performance history for each student observed, consisting of the average of mastery speed, measured in attempts, across all prerequisite skills.

Student	Prerequisite	Mastery Speed	Skill	First Response Correctness
Tom	Adding	4	Division	Correct
Tom	Mult.	8	Division	
Bill	Adding	3	Division	Incorrect
Bill	Mult.	6	Division	
Joe	Adding	3	Division	Correct
Joe	Mult.	3	Division	
Sue	Adding	5 LPC	Division	Incorrect
Sue	Mult.	DNF LPC	Division	

⇨

Attempts	Prediction	#students
3-4	1	1
4-8	0.5	2
8+	0	0
DNF High	0	0
DNF Low	0	1

Figure 7. An example to explain how to generate the five bins of students for a post skill.

Our method creates a table of probabilities for each of five bins, or categories of students, each bin contains the students whose average mastery speeds across all prerequisite skills are in corresponding range, as exemplified in Figure 7. The prediction of each bin is calculated using a training set and represents the number of students of each category that answered the first problem of a subsequent skill correctly. The first bin includes those students that averaged 3 to 4 attempts, inclusively, to master each prerequisite skill; the second bin are students that averaged 4 to 8 attempts exclusively; the third bin, averaged 8 or more attempts; the other two bins contain the students who did not reach mastery status on any prerequisite skills, and are marked as "Did Not Finish (DNF)" category. Since there is a large amount of students following in these categories, so we separate them into two bins according to their percent of correctness across all prerequisite skills: the fourth bin, "DNF High", contains students who have high percent of correctness (greater than or equal to 66.67%); the fifth bin, "DNF Low", with low percent of correctness (less than 66.67%). Thus, bin four represents students that failed to complete the prerequisite skills for reasons other than lack of knowledge, while students in bin five were struggling and perhaps experiencing wheel spinning (Beck & Gong, 2013);

In the example presented in Figure 7, Joe averages 3 attempts to master each prerequisite skill and answers the first problem of the subsequent skill correctly. As such, he is categorized under the first bin, and contributes to the probability of 1.0 due to all students in that bin answer the first question correctly (even there is only one student here). Respectively, since the Tom falls the second bin and answers first question correctly while Bill also in the second bin but has incorrect first response, the prediction of the second bin is 0.5; Sue falls in the fifth bin, according to her response, the prediction of the fifth bin is 0;

3.2.3. Results

Our dataset was comprised of real-world student data from the 2009-2010 academic year taken from the ASSISTments tutoring system. Our method focuses entirely on predicting the first response of each student attempting a new skill. As our dataset was built in a real classroom environment, teachers were responsible for determining which skills to assign, as well as the order to do so. Many such factors could influence the accuracy of our results, so only those skills that fit strict criteria are considered. A skill is only considered if it has one or more existing prerequisite skills; if no data exists for a particular skill, it cannot be viewed as a prerequisite for any skill. Furthermore, only students that exist in all prerequisite skills as well as the subsequent skill are used for our trials.

Table 3. The overall percent of correctness on the first response of five bins..

Bin	#students	%correct on First Response
1	806	61.79%
2	1170	60.00%
3	172	54.65%
4	732	52.59%
5	586	50.51%

Table 4. Result of three approaches.

	RMSE	AUC
Majority Class	0.467	0.673
KT	0.467	0.687
Prerequisite Binning	0.407	0.763

Table 3 shows the distribution of knowledge within each bin across all skills in the observed dataset. The values show a distribution of higher knowledge students in the lower bins and lower knowledge students in the higher bins. This result supports the claim that our method is properly representing the intended level of knowledge.

Comparison of Overall Performance

The results of our method, entitled "Prerequisite Binning" in Table 4, was compared to knowledge tracing as well as a majority class prediction to act as a control in our experiment. The majority class is a prediction made for all students using the average correctness of the dataset. Knowledge tracing was run using Kevin Murphy's Bayes Net Toolbox for MATLAB (Murphy) with initial parameters of 0.30, 0.14, 0.20, and 0.08 for prior, learn, guess, and slip respectively. For our experiment, we ran a five-fold cross validation on our dataset, using 80% of the data from each skill as a training set to predict the remaining 20%.

Each of the three prediction methods are compared using RMSE and AUC two common measurements of error. The results in Table 3 represent the averages of all folds for each method. From this result, the prerequisite binning method outperforms the majority class in both metrics indicating that it is a successful prediction method. When compared to knowledge tracing, however, the results show nearly the same RMSE value, but a superior AUC value. While the binning method may not outperform knowledge tracing in all metrics, the predictive accuracy is comparable.

The purpose of this work is not to provide a method that outperforms KT, but rather to construct a modeling method that can provide teachers with more meaningful information regarding student knowledge. Unlike KT, where the learned parameters such as prior/initial knowledge are unusable metrics in describing true student knowledge due to the identifiability problem, our binning method provides an initial knowledge estimate based on previously observed performance; this initial knowledge estimate, represented as the probabilistic prediction calculated for each bin, is shown to be just as reliable as KT in predictive accuracy, while also providing a more definitive metric to describe a bin-wide initial knowledge that avoids problems of identifiability.

Comparison over Individual Skill

We also compare our method with KT on each individual skill. Figure 8 shows the difference of RMSE for these two models, that is: RMSE (KT) - RMSE (Bin); each positive difference value, therefore, indicates that our binning method outperforms KT in that skill, while negative difference values indicate KT outperforms binning in that particular skill. Each bar in the figure has an accompanying p-value above. This p-value is computed by applying a statistical T-test on the five-fold cross validation results. From this figure, we observe that our method has remarkable improvement on 14 out of 28 total skills. Looking at the T-test results, our method outperforms KT at a statistically significant level (p-value < 0.05) on 3 skills.

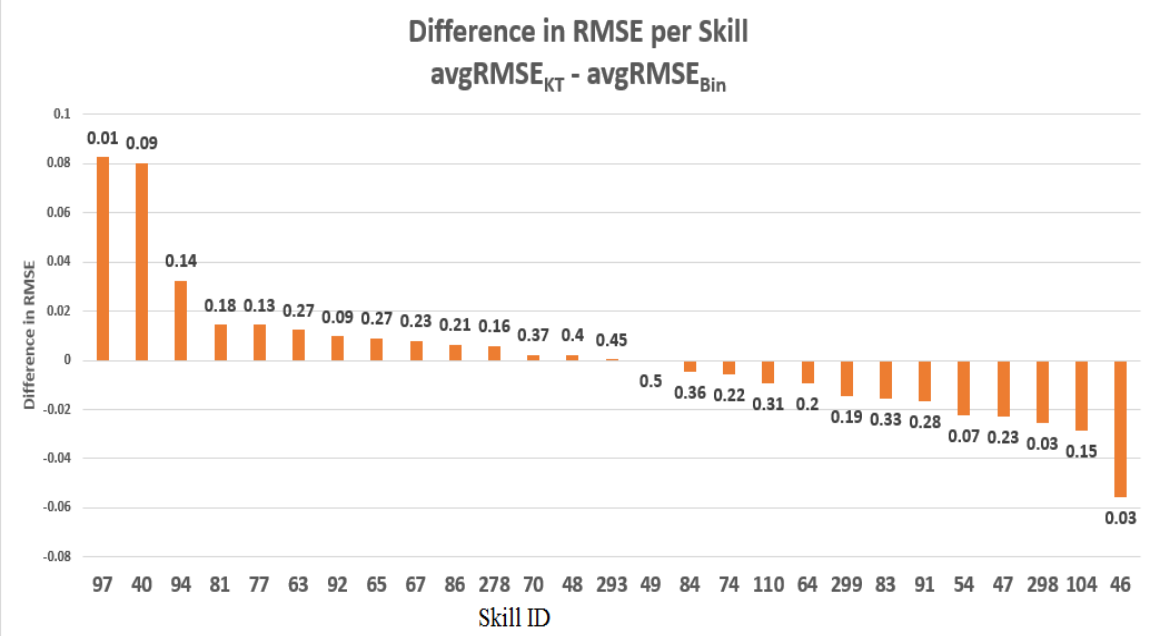


Figure 8. The difference of RMSE per skill when comparing prerequisite binning to standard knowledge tracing. The value above each bar indicates the p-value of the difference.

This result answers the second question in introduction, asking "are all prerequisite relationships reliable?" In accordance with our initial thoughts, the stronger the relationship between a prerequisite and subsequent skill, the better we can predict the performance of the subsequent skill from the knowledge of the prerequisite skill. Therefore, at least on skills 97 and 40, the skills with statistically significant better results, we have strong confidence that the prerequisite relationships are reliable. For those skills with significantly lower results, skills 54, 298, and 46, the causal relation of the prerequisite skills may not be strong as expected by domain experts. All other skills, however, do not illustrate results significant enough to make a claim.

These particular inconclusive results may be explained by inspecting our dataset. Many students, as indicated by our dataset, attempt less than three problems, preventing mastery and also making it more difficult to properly estimate knowledge. There may be two reasons for this occurrence. First, the prerequisite skills may too hard for the students to master. This may result from the teacher's decision not to assign particular prerequisite skills, or the skill relationship graph is incomplete. A second possibility may allude to a case where a teacher does not assign enough questions for students to master the prerequisite skills. As a teacher has control over the administering of skill problems, a number of such scenarios could lead to such results. In summary, these findings potentially indicate a need to further inspect the causal relationships defined by domain experts as they appear in the observed systems.

3.3. Prerequisite Effect in Wheel Spinning Models

3.3.1. Background

Many Intelligence Tutoring Systems (ITS) make use of a mastery learning framework where students continue practicing a skill until they master it. However, some students are unable to achieve mastery despite having numerous opportunities to practice the skill. As a result, these students are stuck in the mastery learning cycle of the ITS and are given additional problems on a topic they are unable to master. These students are referred as "wheel spinning" on the skill (Beck & Gong, 2013). As defined in (Beck & Gong, 2013), a student who takes 10 practice opportunities without mastering a skill is considered to be wheel spinning on this skill.

Beck and Gong (Beck & Gong, 2013) developed a model, consisting of 8 features, to predict which students will wheel spin on a skill. They found that there is a relationship between wheel spinning and gaming the system (Baker, Corbett, Roll, & Koedinger, 2008). Beck and Rodrigo (Beck & Rodrigo, 2014) constructed a causal model (using non-Western students) that situated wheel spinning in the face of affective factors. They found that wheel spinning and gaming were strongly

related. This work also presented a path model that found gaming was not causal of wheel spinning, but rather, wheel spinning was related to a lack of prior knowledge, which in turn led to gaming. A more concrete wheel spinning model is developed in (Gong & Beck, 2015), in which three aspects of features are considered: student in-tutor performance, the seriousness of the learner, and general factors. However, these models do not provide actionable results for how to make a student less likely to wheel spin on a skill, or how to get an already wheel spinning student unstuck.

A natural question is why are some students able to learn a skill and achieve mastery, while other students fail to do so? One plausible hypothesis of what makes wheel-spinning students different from their peers is a difference in ability to learn the skill. Students certainly differ in cognitive abilities, but addressing such would be beyond the scope of most interventions ITS developers can develop. Another plausible difference in ability to learn the skill is due to differences in student preparation. For example, if students do not understand the concept of equivalent fractions, they will have great difficulty mastering the later skill of addition of fractions, which requires them to solve problems such as 1/3 + 1/4.

We define a skill S's prerequisite skills as those skills necessary to be mastered before studying skill S., in We incorporate the prerequisite structure into wheel spinning model, in order to check if prerequisite performance has impact in wheel spinning of post-skills. Although prior research has proposed automatic algorithms of adapting prerequisite structures (Brunskill, 2011; Philip Jr., Cen, Wu, & Koedinger, 2008; Vuong, Nixon, & Towle, 2011), we instead use a prerequisite structure developed by a domain expert.

As an overview, we abstract students' prerequisite performance as a feature, and then add this feature into the wheel-spinning model (Beck & Gong, 2013). Our main points include: 1) determine if there is connection between the prerequisite performance and the wheel spinning of post-skill; 2) explore how prerequisite factor would affect wheel spinning model; 3) compare the prerequisite factor with another possible effect that could cause wheel spinning – students' general learning ability.

3.3.2. Method

The model in our experiments is different from the Beck and Gong's model (Beck & Gong, 2013) in two places: one is that we use one more feature in the model, the prerequisite feature, I will introduce how to compute it in the next few parts; the other is that in some experiments, we treat the feature – prior response count – as a covariate, not a factor like in their model. We found this parameter's affect was approximately linear, and thus treating it as a covariate made more sense.

We call the model based on these 9 features the baseline model, and compare it with a model that includes the prerequisite performance.

Compute Students' Performance on Skills

In this work, our goal is to find the influence of students' prerequisite performance on wheel spinning. So the first step is to choose which measure to represent students' performance on each skill. In this work, we regard a student's percentage of correct responses to questions involving a skill to be his performance on that skill.

However, a student could answer correctly, by chance, even though this student does not understand the skill at all. Similarly, a student could give the wrong answer through a careless mistake, as in the guess and slip parameters in the Knowledge Tracing model (Corbett & Anderson, 1994). These two cases will deviate the student's performance from his/her "true understanding" on the skill, especially if the student has very few practices. To deal with these cases, we balance the "accidental performance" with student's overall performance on all skill. The formula for calculating a student's performance on a skill i is:

$$P_i = \frac{1}{2^x} * \overline{R} * S_i + \left(1 - \frac{1}{2^x}\right) * C_i$$

- x: The number of practices on this skill;
- S_i: The percent correctness of skill i, $S_i = \frac{\#\text{correct practices}}{\#\text{overall practices}}$ (over all students). This also reflects the hardness of skill S_i.
- C_i: The student's percent correctness on skill i, $C_i = \frac{\#\text{correct practices}}{\#\text{overall practices}}$ (over the student st_1).
- $R_i = \frac{C_i}{S_i}$: This represents how well the student st_1 does on skill i comparing with the other students.
- $\overline{R} = \frac{\sum_{i=1}^{m} R_i}{m}$: m is the number of the student's started skills.

Notice in the formula, the more practices on a skill, the more weight is assigned to the performance on this skill. Take the data in Table 5 as an example. There are in total 4 trials for skill s1, of which 3 are answered correctly, so its correctness is 0.75. The correctness of the other two skills is: s2, 1.0; s3, 0.5. The student, st1, answered two problems of s1, getting one correct and the other incorrect. So this student's correctness of s1 is 0.5, and $R_1(st1) = \frac{0.5}{0.75} = 0.67$. We can also get that $R_2(st1) = 1.0, R_3(st1) = 0$, then $\overline{R}(st1) = 0.56$. Hence, the student st1's estimated

understanding on the skill s1 is: $\frac{1}{2^2} * 0.56 * 0.75 + \left(1 - \frac{1}{2^2}\right) * 0.5 = 0.48$. All the performance results are shown in Table 6. Sometimes, a student's adjusted performance is larger than 1, as the student st2's performances on skill s1 and s2. This effect can occur by a student doing very well on a very difficult skill. In this paper, we normalize the values to bring them in the range from 0 to 1.

Table 5. A small sample of students' practices.

Student	**Skill**	**Problem**	**Correct?**
st1	s1	p1	1
st1	s1	p2	0
st1	s2	p3	1
st1	s3	p4	0
st2	s1	p1	1
st2	s1	p2	1
st2	s3	p5	1

Table 6. Calculated skills' hardness and students' performance according to the data in Table 5.

Skill	Correctness	Student performance		Normalized performance	
		st1	st2	st1	st2
s1	0.75	0.48	1.06	0.45	1
s2	1.0	0.78	1.67	0.47	1
s3	0.5	0.28	0.92	0.3	1

Compute Prerequisite Performance

Once the normalized students' performances have been computed, the next step is to think about how to represent prerequisite performances, and then incorporate it into the wheel-spinning model. If a skill has only one pre-required skill, such a representation is straightforward: the student's adjusted performance on that pre-required skill. But what if a skill has multiple prerequisites? In our data set, 39 out of 128 skills have multiple prerequisites. There are a variety of approaches for handling multiple prerequisites. We chose two different methods to compute the prerequisite performance: weakest link and weighted by hardness.

1) Weakest link

This method regards the prerequisite skill with the worst performance, called weakest link, to have strongest connection with post-skill's wheel spinning. Thus, it uses the lowest performance value in all prerequisite skills as the wheel-spinning model's input for prerequisite performance. For example, in Table 5, if skill s1's prerequisite skills are s2 and s3, then the prerequisite performance for student st1 on skill s1 is estimated as 0.3 (normalized).

2) Weighted by hardness

In this method, we sum up a student's prerequisite performances by assigning a corresponding weight to each prerequisite skill, according to the skill hardness. We assume that the harder a prerequisite skill is, the more importance it has. Here we define a skill's hardness to be $1/correctness$. Thus, for a skill, the representation for its prerequisites is calculated as:

$$\mathrm{Pr}_i = \frac{\sum_{j=1}^{n} w_j P_j}{\sum_{j=1}^{n} w_j}$$

- n: Number of prerequisites.
- P_j: A student's performance on the jth prerequisite.
- $w_j = \frac{1}{S_j}$: The weight assigned into the jth prerequisite. S_j is the correctness of this prerequisite.

We also suppose that the skill s1's prerequisites are s2 and s3, then using the data from Table 6, the student st1's prerequisite performance on skill s1 is 0.36; respectively, the student st2's prerequisite representation value for s1 is 1.

Define General Learning Ability

Our approach is to construct a variable, which we refer to as General Learning Ability (GLA) that encapsulates some of the constructs like diligence, home support, raw ability, and so on. GLA refers to a student's latent ability that affects his ability to learn new skill, similar in spirit to the one dimensional trait in Item Response Theory (IRT) (Embretson & Reise, 2013). In IRT, a student's trait is assumed measurable; it is measured through a series of adaptive questions given by a tutoring system.

To simplify our work, we measure student's general learning ability as following steps:

a) For each student-skill pair, randomly select the other two started skills. Here a started skill means the student has practiced at least one problem on it;
b) Compute the performance values for the two skills, as described in the Method part;
c) Take the average of those two performance values as the general learning ability for this student-skill pair.

Our intuition in defining GLA in this manner is that if the reason for WH's strong gradient with wheel spinning is due to the knowledge of the prerequisite being important, we would expect GLA

to perform poorly. However, if the power of WH comes not from estimating a particular aspect of student knowledge, but rather than providing a proxy measurement for a student's general ability and willingness to learn, we would expect estimating the student's knowledge of two random skills would work as well. We chose to use two random skills since that was the average number of prerequisites, and we wanted to avoid issues with one measure having lower variability (and hence higher reliability) simply by being an aggregate of more skills. One potential drawback of our approach is that two skills is a small number, and in some cases will certainly provide an over- or under-estimate of knowledge for a particular student. However, since our sample size is large enough, 48256 student-skill pairs in total, this approach is unlikely to produce skewed results.

3.3.3. Results

Data Set

The data in this work is from ASSISTments. We tracked all ASSISTments students when they used the system to practice Math problems for almost a full year from September 2010 to July 2011. This data set contains 7591 different students, and we randomly select 4976 of the students (about 2/3 of students) to form our training data set, while the other students comprise the testing data. There are 31301 student-skill pairs in the training set and 16955 in the testing set. In this work, we consider students who fail to achieve mastery within 10 practice opportunities for a skill (including indeterminate cases (Beck & Gong, 2013) as wheel spinning, which results in 20.6% instances in the training set as wheel spinning and 19.2% in the testing set.

In the training data, there are 177713 problems solved by the students, while 97768 problems in testing data. These problems cover 128 different skills. In the training and testing set, students learn different skills. The maximum number of learned skills by a student is 61, and the average is 6.4. As aforementioned, the prerequisite-to-post skill structure is defined by domain expert as a recommended sequence of topics for instructors. Among the skills in our data set, 66 skills have at least one prerequisite. Some skills have multiple prerequisites, the max number of prerequisites is 8, and the average is 2.4.

However, it is the teacher's choice which skills and in which order to assign to students. Consequently, the majority of student-skill pairs do not have any started prerequisite skills in our data set, as shown in Figure 9. Apparently (and understandably), teachers are less likely to assign review material than to focus on new topics. The maximum number of started prerequisites is 4, and the average is only 0.37. Thus, our experiments will run over three different data sets:

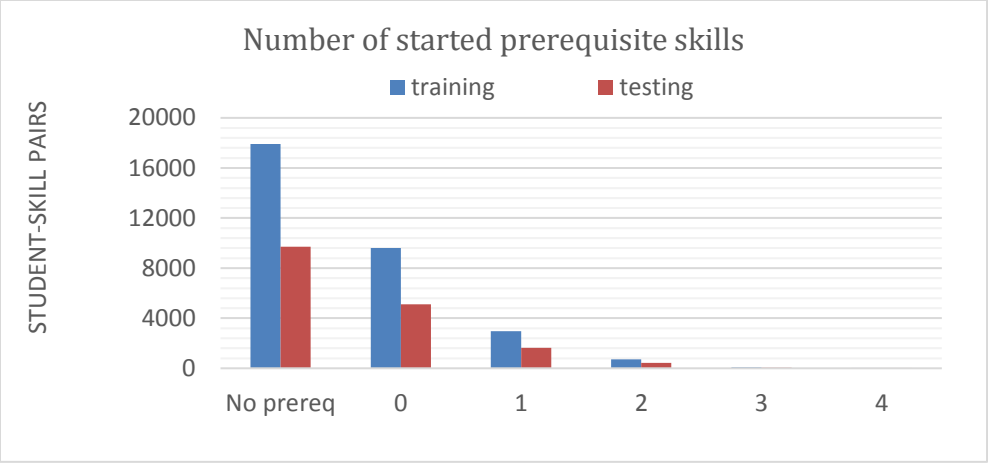


Figure 9. Distribution of number of started prerequisite skills in training set and testing set.

a) D1: the whole data set, as depicted in Figure 9, which is split into training and testing set.

b) D2: the prerequisite data set. This data set excludes the skills that have no prerequisite skills, as identified by the domain expert, from D1. Thus, it is comprised of the points on the x-axis in Figure 9 corresponding to 0, 1, 2, 3 and 4. It is also split into training and testing set, and its training set is constructed from the training set in D1 by removing the non-prerequisite skills, while its testing set from testing set in D1 respectively.

c) D3: the started prerequisite data set, and includes only student-skill pairs where the student has at least begun one of the prerequisites. This data set excludes the skills that have no started prerequisite skills from D2. Thus, it is comprised of the points on the x-axis in Figure 9 corresponding to 1, 2, 3 and 4. Similarly, its training (testing) set is generated from training (testing) set in D2 by removing non-started-prerequisite skills.

The reason for these three datasets is that they answer different research questions. D1 enables us to investigate the impact of prerequisite performance on wheel spinning in an already-existing system in a real-world deployment. That is, how much benefit would we see in the current usage context of the tutor. Unfortunately, that real-world deployment involves teachers assigning no work on most prerequisites, and thus no information about student prerequisite knowledge is available to the model. D2 enables us to examine where there is at least potential benefit. D3 enables us to answer questions about whether a system that had fuller information about prerequisite would perform better at detecting wheel spinning. D3 lets us consider possible changes to policy where teachers are more willing to assign review work, or a system is better able to access past student performance to assess prior knowledge.

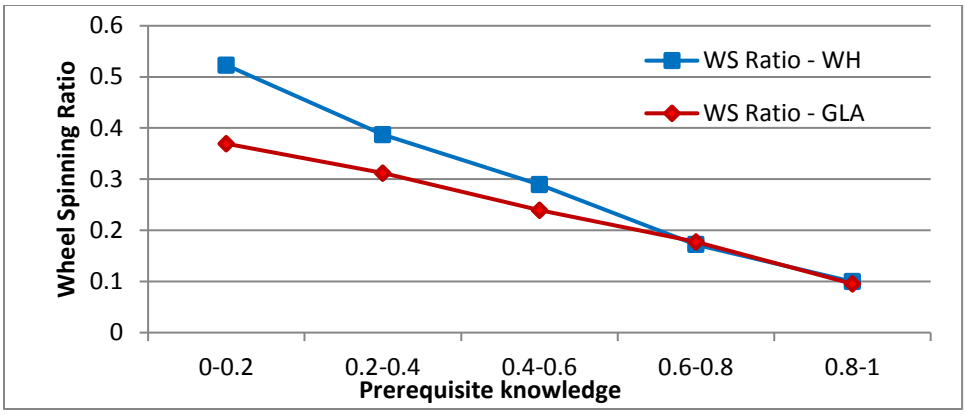


Figure 10. Wheel spinning ratio with respect to prerequisite knowledge and general learning ability on the data set D3.

The Gradient of Wheel Spinning Ratio

In order to determine how likely a student will be to wheel spin on a skill based on his corresponding prerequisite performance value, we focus on the training set of D3. We separate D3 into 5 bins according to the prerequisite performance value, calculated by the method weighted by hardness. The wheel spinning ratio in each bin is shown in Figure 10, named WS Ratio - WH.

As observed in the figure, there is a strong gradient with respect to the prerequisite performance: students in the bottom 20% of pre-required knowledge exhibited wheel spinning behavior 50% of the time, while those in the top 20% of pre-required knowledge exhibited wheel spinning behavior only 10% of the time. This expresses strong evidence supporting our hypothesis that student's wheel spinning on post-skill results from poor preparation for future learning in terms of prerequisite knowledge.

Changes in Models

To test the impact of prerequisite features, we integrated them into the wheel-spinning model described previously. We compare the effects of different factors in the wheel spinning model, Weakest Link (WL), Weighted by Hardness (WH), and General Learning Ability (GLA). Table 7 shows the results of training each model on the training test, and evaluating it on the test set.

In this experiment, we use the Cox and Snell R square (Hosmer Jr & Lemeshow, 2004) and AUC (area under curve) to measure model fit. As we can see, the model does not appreciably change in the data set D1, because the part of the data containing started prerequisite skills is such a small component of the data. In D2 and D3, the model is improved slightly by integrating the prerequisite feature, WH or WL. This result supports that prerequisite performance is useful in determining students' wheel spinning status in post-skills. We can also notice that the model with GLA has the similar results with the ones with WH and WL.

Table 7. Measurements of different models.

Model	R Square			AUC		
	D1	D2	D3	D1	D2	D3
Baseline	0.285	0.301	0.264	0.879	0.888	0.884
Baseline +WL	0.285	0.302	0.268	0.879	0.889	0.887
Baseline +WH	0.285	0.302	0.268	0.879	0.889	0.888
Baseline +GLA	0.291	0.306	0.268	0.883	0.891	0.887

Table 8. P-values of paired t-test. For each pair of models, the t-test is applied on the test results of each data set (D1, D2, D3).

	Baseline	Baseline + WL	Baseline + WH
Baseline + WL	<0.01,<0.01, <0.01		
Baseline + WH	<0.01,<0.01, <0.01	0.42, 0.40, 0.74	
Baseline + GLA	<0.01,<0.01, <0.01	<0.01,<0.01, 0.52	<0.01,<0.01, 0.49

Furthermore, to compare the difference between models, a paired t-test is applied on the results of each pair of models, as shown in Table 8. The result shows that adding a factor - WH, WL, or GLA – into the baseline model makes it performing significantly differently in all data sets, D1, D2, and D3. On the other hand, the model "Baseline+WH" and "Baseline+WL" have the similar results in those three data sets, which also implies these two prerequisite features have similar effect in the wheel spinning model. More interesting, the p-values indicate that the model with GLA and the model with WH (or WL respectively) are significantly different in D1 and D2, but not in D3. Since the GLA factor is defined as the average performance of two randomly selected skills. In the data set D3, every student-skill pair is linked with at least one prerequisite skill. Thus, it is very likely some GLA values are constructed from prerequisite skills, which makes the two models similar in D3.

Impact of Prerequisite Effect on Predictive Models

We now move to determining the impact of the prerequisite feature on the predictive model. In our intuition, the prerequisite factor might have strong effect in predicting wheel spinning when a student just starts learning a post-skill, and the effect weakens with time as the student solves problems on the post skill.

In the logistic regression algorithm, researchers typically use the odds ratio, exponential the coefficient, to represent effect of the corresponding feature (Hosmer Jr & Lemeshow, 2004). Then the coefficient could be also used to represent the effect on the model. Therefore, in this work, we use the coefficient of prerequisite feature to reflect its effect in predicting students' wheel spinning on post-skill.

In this experiment, we group the D3 of training set by amount of practice on the skill, and construct a wheel spinning model for each group. The coefficients of prerequisite feature (for the WH model) in the corresponding models are shown in Figure 11. As we can see, the coefficient representing the impact of prerequisite knowledge has the highest value at the beginning, and it decreases in influence as students obtain more practice on the skill. This result support our intuition that the prerequisite factor is a good predictor for wheel spinning only at the beginning stage of learning post-skill. Thus, prerequisite knowledge is useful for overcoming the cold start problem in student modeling. When a student first starts working on a skill, his performance on that skill provides little basis with whether to classify him as likely to wheel spin or not. In this situation, knowing how he performed on the prerequisite skills provides some information in his ability to master the current material. As the system observes more and more performances on the skill, those performance provide a much more pertinent source of information about the student's likely trajectory, and the relative importance of prerequisite skills diminishes.

The decrease in in predictive performance for the WH coefficient is monotonic and roughly linear. From a standpoint of statistical significance, the WH coefficient is reliably different than 0 for practice opportunities 1 through 7 ($p=0.026$ at the 7th opportunity). At the 8th opportunity, the impact of the WH coefficient has $p=0.51$.

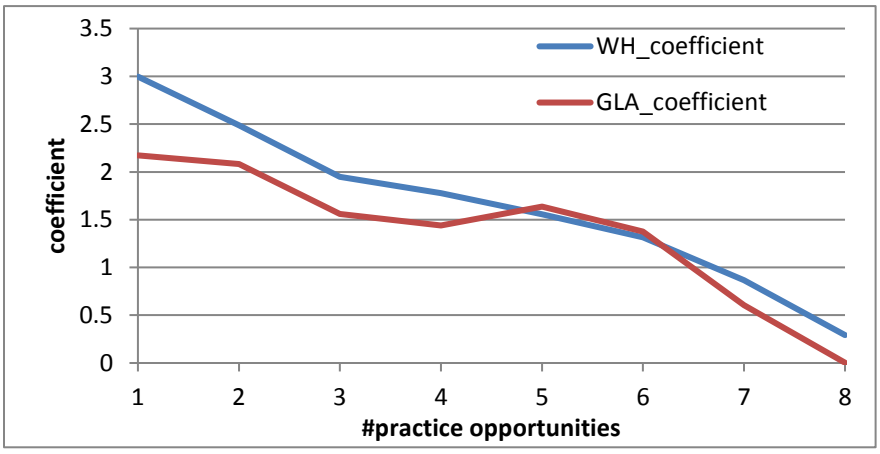


Figure 11. The changes of coefficient with respect to number of practice opportunities on the data set D3.

Understanding What Prerequisite Performance Really Represents

The performance of the WH feature raises an interesting question: to what does it owe its predictive power. Although we refer to this feature as representing student's prerequisite knowledge, it captures much more than just knowledge. For example, if one student demonstrates strong performance on prerequisite skills and the other does not, those students probably differ in many dimensions beyond knowledge of the skill: diligence in doing math homework, support at home, raw ability at learning new concepts, and perseverance when stuck. Wrapping this bundle of constructs together and calling it "prerequisite knowledge" certainly simplifies discussion, but does a disservice to accuracy. Therefore, we perform a baseline experiment to investigate what prerequisite knowledge represents.

Since the effects of two prerequisite features, WL and WH, are pretty much the same in the wheel spinning model. Therefore, we will compare only the WH with the GLA. These two features are compared though three different experiments.

The first experiment is to construct wheel spinning ratio gradient for GLA. As we can see in Figure 10, there is the same broad trend for both GLA and WH. For both measures, students with lower general learning ability are more likely to be wheel spinning, which is in accord with our common sense. By comparing the two wheel spinning ratio gradients, we notice that the ratio is the same when the WH and GLA values are high; that is, if a student's performance is relative high (> 0.6) for WH and GLA, then there is a similar chance the student will wheel spin. However, in the lower range of 0 to 0.6, students are more likely to be wheel spinning according to WH value than the students having the same GLA value. This result suggests that prerequisite factor has stronger correlation with wheel spinning than general learning ability, although general learning ability has strong overlap.

The second experiment is to add the GLA into wheel spinning model and compare the model measurements. According to the results in Table 7, adding the GLA into the baseline model makes more improvement than adding the WH on the data set D1 and D2. This is because the student-skill pairs with pre-required knowledge are very rare in those data sets, while every student-skill pair is assigned with a computed GLA value based on that student's performance on a pair of random skills. The model with GLA and the model with WH on the data set D3 have nearly identical performance.

The third experiment is to compare the effect in the learning procedure. As seen in , the GLA coefficient also decreases with respect to the number of practice. But in the first 5 practices, the

slope of GLA coefficient is more moderate than the slope of WH coefficient, which defends the statement that the prerequisite factor is useful in predicting wheel spinning at early learning stage. By examine the GLA coefficient Wald statistic p-value, it is also statistically reliable ($p<0.05$) before the 7th practice.

3.3.4. Discussion

Prerequisite Structure

As aforementioned, the prerequisite structure used in this work is defined by domain experts. Through this structure, the experts suggest a general curriculum over all grades, not specified in a single year or a single class. It is certainly possible that our structure is in error either by missing some links and incorrectly creating others. Such errors would impact the results.

Moreover, in the method of computing prerequisite performance for a post-skill, we assume that the prerequisite skill with the worst performance (or the hardest prerequisite skill) has the strongest influence in learning post-skill. However, this assumption might be inappropriate here. Botelho et al. also illustrate in their experiments that the prerequisite relation in some post-skills are not as stable as expected by domain experts (Botelho, Wan, & Heffernan, 2015).

Therefore, there are two possible ways of improving our experiments. The first one is to construct a prerequisite structure specifically for the data. Previous works have been focused on this area. For example, Vuong et al. introduce a method for finding prerequisite structure within a curriculum (Vuong, Nixon, & Towle, 2011). Their method calculates the overall graduation rate for each unit, and regards Unit A as prerequisite knowledge for Unit B if the experience in Unit A promotes graduation rate in Unit B.

The other possible way is to measure the correlation between each prerequisite skill and a post-skill, and then we can obtain which prerequisite skill is most effective in affecting learning post-skill. Vuong et al. also distinguish the prerequisite relationship between significant and non-significant in their work (Vuong, Nixon, & Towle, 2011).

Prerequisite-post Relation

Obviously, students' general learning ability influences their performance in both prerequisites and post-skills. Therefore, one might argue that there is no direct causal prerequisite-post relationship. The student who is wheel spun on learning post-skill as well as lack of pre-required knowledge is mainly because he/she has weak learning ability, as shown in Figure 12. In this view, GLA is the primary driver of both prerequisite and post-performance.

According to this argument, a consequent case would be: a student who is wheel spun on a skill, he/she will be wheel spun on every skill, due to the weak learning ability. However, in our data set, the wheel spinning ratio of the students who have at least one wheel spinning case is about 23%. Thus, the GLA is an effective factor in wheel spinning, but not a unique or crucial one. Another drawback of this model is that, for low levels of performance, prerequisite knowledge is more strongly related to wheel spinning than GLA. Therefore, even if GLA is the primary driver, there is apparently some impact of prerequisite knowledge on post-performance, represented by the dotted line in Figure 12.

In order to validate the structure in Figure 12, a subtler model should be constructed, in which students' GLA is finely measured. A proper way is to utilize the IRT model to estimate a student's trait; this trait is regarded as the GLA value. And then it is used in predicting if the student will be wheel spinning or not. Meanwhile this trait is updated for each item practiced or for each skill learned. The similar work is in (Huang, González-Brenes, & Brusilovsky, 2014), the authors integrate temporal IRT into Knowledge Tracing model, in order to track students' knowledge stage and predict next problem correctness.

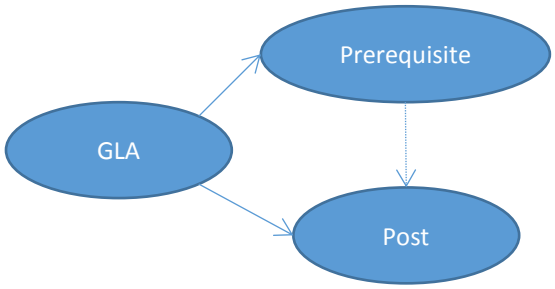


Figure 12. A structure to explain indirect prerequisite-post relationship.

CHAPTER 4

K-armed Bandits

4.1. Introduction

In the k-armed bandit problem (Robbins, 1985), an agent has to select an action from n possible options, at each time step. After performing the selected action, the agent receives a reward derived from an unknown and action-specific distribution. Its objective is to maximize the sum of (possibly discounted) rewards over time. Since the reward distributions are initially unknown, the agent should explore different actions to learn each action's distribution. However, an agent cannot explore and gather information indefinitely, as it is also trying to maximize its total performance as indicated by the reward function. This tradeoff of exploration vs. exploitation is well known in machine learning (Audibert, Munos, & Szepesvári, 2009).

As aforementioned, this dissertation work is aiming at using k-armed bandit strategies to provide students with optimal tutoring interventions in ASSISTments. In more details, the problem is defined as: each student is assigned, one by one, with an appropriate intervention, based on previous students' choices and corresponding received rewards. The objective is to maximized the total rewards over all students. But why do we need k-armed bandits in ASSISTments?

Table 9. Example of students' performances in different conditions.

Condition	Mean (correct)
1	0.70
2	0.75

Take a real experiment in ASSISTments as an example. In this randomized controlled experiment, students are randomly assigned into two different conditional problem sets to learn mathematic skills, one in which the problem presentation and hints suggest a spatial approach and one in which the problem with an analytic approach. After learning in the problem sets, students face the same posttest. Students' performances in the posttest are shown in Table 9. As we can see in this table, with random assignments, the average correctness in the posttest over all students is about (0.70+0.75)/2 = 0.725. However, if we use k-armed bandits in this experiment, since students in the analytic problem set perform better than in the spatial problem set, students are more likely to be assigned into the analytic problem set. Then the average percent of correctness is approaching 0.75, because we have to explore to collect performance in spatial condition. Therefore, applying the k-armed bandits in the ASSISTments could bring better students performances.

In this chapter, I will talk several related works and their weakness, and then discuss whether it is worthwhile to apply fancy or complicate algorithms in the system, by comparing them with a single selection strategy based on statistical t-test. Finally, I will analyze the results of data sets from two ASSISTments experiments.

4.2. Related Works

K-armed bandit algorithms have been widely applied in various areas, such as web search (Feng, Heffernan, & Koedinger, 2009; Radlinski, Kleinberg, & Joachims, 2008), internet advertising (Babaioff, Sharma, & Slivkins, 2009; Li, Chu, Langford, & Schapire, 2010), queuing and scheduling (Liu & Zhao, 2010), and education (Clement, Oudeyer, Roy, & Lopes, 2014; Clement, Roy, Oudeyer, & Lopes, 2015; Liu, Mandel, Brunskill, & Popovic, 2014; Silva, Direne et al., 2015).

A crucial issue in the applications is to demonstrate usefulness of algorithms. A common approach is to compare the k-armed bandit algorithms with a random selection strategy or predefined sequences of actions (Silva, Direne et al., 2015; Wang, Wang, Hsu, & Wang, 2014). Typically, the bandits give stronger performance than the baseline algorithms, which is expected as the bandit algorithms are using more information. The baseline approaches are just designed to show how much better the bandit is doing than a simple approach. Other works use some simple k-armed bandit algorithms as a baseline, such as ε-greedy (Pavlidis, Tasoulis, & Hand, 2008; Vermorel & Mohri, 2005; Wang, Wang, Hsu, & Wang, 2014).

In this work, we propose another selection strategy that is based on statistical t-test called Strawman and we deploy it in educational experiments. As our work involves deciding which experimental condition to assign a student, when a particular experimental condition is less effective, we chose t-tests as our mechanism for excluding actions. We named this technique "Strawman" as t-tests were developed in 1908, and are hardly cutting-edge methodology. Our intention was to create a plausible lower-bound on performance. The Strawman strategy randomly chooses one of the possible actions at each time step. After it observes a reward, it then compares each action against all of the other actions; any action that is statistically reliably worse ($p < 0.05$) than any other action is dropped from future consideration. Once a single action remains, it considers that action as the unique option. Therefore, there are two phases in this strategy. In the first phase, it employs random selection and in the second phase it uses a greedy selection. Langford and Zhang propose a similar method, epoch-greedy, in which a random selection step is followed by absolute exploitation steps in each epoch (Langford & Zhang, 2008). The difference is that the Strawman would never do

exploration again once it detects the reliably "best" action, but the epoch-greedy method would select an action randomly in other epochs.

Another issue in evaluation of the k-armed bandit algorithms is about cost. Since deploying the evaluations in real environments is time consuming and sometimes money consuming. To resolve this issue, Li et al. deploy the evaluation on the collected offline records (Li, Chu, Langford, & Schapire, 2010), in other works, like in (Clement, Oudeyer, Roy, & Lopes, 2014; Pavlidis, Tasoulis, & Hand, 2008), the algorithms are evaluated on the simulated instances. In this work, we will evaluate the Strawman algorithm and other k-armed bandit algorithms on the data sets that contain students' responses in two ASSISTments experiments, through simulating applying them in real environment based on random sampling with replacement.

4.3. K-armed Bandit Algorithms

We will compare three k-armed bandit algorithms ε-greedy, simulated annealing, and UCB1, against Strawman, based on statistical t-tests. the probability of selection each action at each step based on an action-value function, called Q function; while UCB1 selects an action with the largest estimated upper confidence bound.

Given the action set: $A = \{a_1, a_2, \dots, a_n\}$, the Q function, $Q(a)$, which denotes the expected reward by choosing the action a at current step. A commonly used method to compute the Q function is exponential average. At the beginning, the value $Q_0(a)$ is set as a default value. After each time choosing an action a and receiving a corresponding reward $r_{a,t}$, the value is updated as:

$$Q_t(a) = Q_{t-1}(a) + \alpha(r_{a,t} - Q_{t-1}(a))$$

The constant α is called the stepsize parameter, and represents how quickly the Q-values are updated. This parameter is bounded $0 < \alpha < 1$, where values close to 1 indicate rapid updating, but possible instability and non-convergence. Values near 0 converge slowly, but are much more predictable in their behavior. A value of $\alpha = 0.1$ is reasonably common.

1. The ε-greedy algorithm selects a random action from the action set A with a fixed probability,$0 \leq \varepsilon \leq 1$, and selects the best action, according to the Q function, with the probability $1 - \varepsilon$.

$$\pi = \begin{cases} random\ action; & if\ \xi < \varepsilon \\ argmax_{a \in A}\ Q_t(a); & otherwise \end{cases}$$

2. In the simulated annealing algorithm, also called decreasing softmax (Cesa-Bianchi & Fischer, 1998), the probability of selecting an action is ranked and weighted according to the estimated value with Gibbs distribution:

$$\pi(a) = \frac{e^{\frac{Q_t(a)}{\tau_t}}}{\sum_{b \in A} e^{\frac{Q_t(b)}{\tau_t}}}; \qquad \tau_t = \frac{1}{t}$$

Where τ_t is a positive parameter, called temperature, and decreases step by step. This algorithm tends to choose explorative actions at early stage, and since the temperature is decreasing, it becomes more and more exploitative as keep playing actions.

3. UCB1, as a member of UCB family proposed by Auer et. al (Auer, Cesa-Bianchi, & Fischer, 2002), plays each action once initially. Afterward, at time t, it selects an action with the largest value as follows:

$$\arg\max_{i=1\ldots n}\left(\overline{r_i} + \sqrt{\frac{2\ln t}{s_i}}\right)$$

Where $\overline{r_i}$ is average of past rewards by playing action a_i, and s_i is number of times playing a_i.

4.4. Methodology

4.4.1. Strawman

In this work, we construct a simple selection algorithm based on the statistical t-test, called Strawman. This algorithm can be considered as a baseline for other k-armed bandit algorithms. We also compare it with the three aforementioned k-armed bandit algorithms by simulating them on two ASSISTments experiments. The Strawman algorithm works as followings:

1. Initially, the action set A contains all possible actions.
2. At the time t, all remaining actions in the action set have the same chance, and the Strawman just randomly selects one from them.
3. When obtaining a reward by playing an action a, record the reward in the a's reward history, e.g. $(r_{a,1}, r_{a,2}, \ldots, r_{a,k_a})$.
4. Compare the action a with every other remaining action in the set by conducting a t-test on their past rewards, e.g. on $(r_{a,1}, r_{a,2}, \ldots, r_{a,k_a})$ and $(r_{b,1}, r_{b,2}, \ldots, r_{b,k_b})$ (for every $b \in A \text{ and } b \neq a$).

5. If there is an action d is significantly worse than any other action according to the t-test – that is, p-value from the t-test is less than a threshold, then remove d from A. Loop to step 2. In this work, the significant level is set to be $p < 0.05$.

For example, in the data set as shown in Table 9, the initial action set is {Condition 1, Condition 2}, if we detect that the action "Condition 1" is significantly worse than "Condition 2" at time t_i, then we drop the former action, and the remaining set is {Condition 2}, and students will be always assigned into Condition 2 afterwards.

4.4.2. Simulation

Evaluating k-armed bandit or other action selection algorithms in real time is costly, it might take a few days, or weeks, or even months to obtain the results. In the meantime, students are being assigned to experimental conditions inefficiently. Consequently, student learning is lower than it could be if a more efficient experimental strategy were used. This issue of inefficient allocation of students raises ethical issues, and we should strive for more benign ways to evaluate competing approaches. An alternative approach is to run the algorithms on a faked data set in which the received rewards are generated from certain distributions, such as normal distributions (Vermorel & Mohri, 2005). However, through this evaluation approach, which algorithm does better depends on specific distributions of the various bandit levers. Besides, this approach does not reveal how selection algorithms perform in the real environment.

Instead of conducting the evaluation on simulated reward distributions, Li et al. (Li, Chu, Langford, & Schapire, 2010) construct a policy evaluator that utilizes the available offline data that was collected at a previous time. But in evaluation process, the historical records with action different from the one selected by a policy are abandoned, which makes the approach improper when the offline data is not large.

In this work, we also evaluate the action selection algorithms on the collected offline data based on random sampling method. Our goal is to measure the performance of a selection strategy, π, at each step of choosing action based on previous activities. The reward corresponds to the selected action at each step is simulated by randomly picking a record from the whole data set that has the same action. The evaluation process is described in Figure 13. Essentially, this simulation process uses bootstrapping (sampling with replacement). Whenever an action is tried, the approach randomly selects a prior student who was given that action, and that student's performance is used as an estimate.

```
Simulation of a strategy π on a set of students D
    do
        randomly pick a student s1 from D
        obtain the action a according to π: π(s1)->c
        randomly pick another student s2 from D with the same action a
        treat s2's reward, r2, as s1's
        update π with r2
    until meet end condition
```

Figure 13. Process of simulating selection strategy on an offline data set.

4.4.3. Metrics

The first metric used in measuring the performance of selection algorithms in this work is the average reward. This metric is commonly used in the k-armed bandit problems, and it represents how a selection algorithm perform at each time. At the time t, the average reward is calculated as: $\frac{1}{t}\sum_{i=1}^{t} r_i$, where r_i is the reward received in the time i.

The second metric indicates whether or not a selection policy is making an explorative choice at each step in the simulation process. To be consistent in the three algorithms in this work, ε-greedy, simulated annealing, and UCB1, we define a policy is making an explorative choice if it selects an action that does not have the best Q value at current step. While the Strawman algorithm is exploring when there are more than one possible actions at each step, since it just randomly selects one from all the remaining actions.

4.5. Experiments

4.5.1. Data

ASSISTments supports researchers to design randomized controlled experiments to validate the effect of one or more factors in tutoring students. The first data set in this work is collected from the experiment in which students are randomly assigned into one of two skill builders to learn a skill. These two skill builders are different in the condition of problem description, marked as condition 1 and condition 2. With completing the skill builder, students are faced with the same posttest.

In the second data set, the experiments conditions are not Skill Builders but instead students are given a set number of items to answer. What distinguishes the three conditions is the type of feedback provided when the student makes an error. In the first condition called “Correctness Only” the students would see a hint button labeled with “Show Answer” that if clicked would tell the student the answer with no explanation at all. The two other conditions were implemented with what ASSISTments called “scaffolding questions.” Scaffolding questions are invoked as soon as the student makes a wrong attempt or if they click on the help button (instead of being labeled as

"Show Answer" as in the first condition, is labeled "Break this problem into Steps") that indicates to the student that if they ask for help they will be given a series of (at least one) questions to help them through the problems. The two conditions labeled "with image" and "no image" both present the student with the scaffolding question that is meant to help them complete the problem. The two conditions differ from each other only by the fact that one condition has an image that was designed to help the student understand. These two conditions differed from the "Correctness Only" in that the correctness only was "help on demand" as opposed to "help given on first error." The details of this study are not actually relevant to this paper, and this paper's method can be applied to any experiment, but if you want you can see details this study was published in (McGuire, Logue et al., 2016).

Table 10. Students' posttest scores in each conditional problem set in the first data set. Students with the bolded condition have higher mean posttest.

	#Students	Mean	Std.
Condition 1	232	0.695	0.332
Condition 2	**237**	**0.746**	**0.320**

Table 11. Students' posttest scores in each conditional problem set in the second data set. Students with the bolded condition have the highest mean posttest

	#Students	Mean	Std.
Correctness only	74	0.518	0.214
With image	**63**	**0.544**	**0.229**
No image	69	0.542	0.201

When applying action selection strategies over the data sets, conditions connect to the problem sets can be considered as actions. Therefore, action set is {Condition 1, Condition 2} in the first data set, and {Correctness only, No image, With image} in the second data set. In this work, we regard students' posttest score as the rewards. According to the students' posttest scores shown in Table 10, the better condition in the first data set is "Condition 2", from Table 11, the best condition is "With image" in the second data set. The purpose in our experiments is to determine how many simulated students each bandit algorithm would have to see to draw those conclusions. In this way, we can experiment and see how differentiable the conditions are after (for example) 500 students' performances have been observed.

4.5.2. Comparison of Mean Rewards

A perfect selection strategy chooses the best action at each time step, so the upper bound of reward is the reward corresponding to the best actions. For example, the upper bound of average reward is 0.746 (Condition 2) in the first data set, and 0.544 (condition "No image") in the second data set.

Respectively, the lower bound is the reward with choosing the worst action. However, if a selection strategy is effective, it should perform better than, at least as well as, the random selection strategy. Therefore, in this work, we consider the reward output from random selection strategy as the lower bound. Regardless of number of students in each conditional problem set, the lower bound of reward in the first data set is: $(0.746 + 0.695)/2 = 0.721$, and 0.535 in the second data set.

In this work, the parameters set for the algorithms are: $\varepsilon = 0.1$ for ε-greedy; the significant level is set to be 0.05 in Strawman; the initial estimation: $Q_0(a) = 1$, and $\alpha = 0.1$ for each condition. As we can observe from the figures, the rewards of those algorithms are close to each other after 200 students, and the UCB1 algorithm performs a little bit worse than others. These results are all better than the lower bound, but do not reach the upper bound. This might be contributed to two reasons: some bandit algorithms continue to explore and so select suboptimal actions, and some algorithms incorrectly converge on the wrong best action.

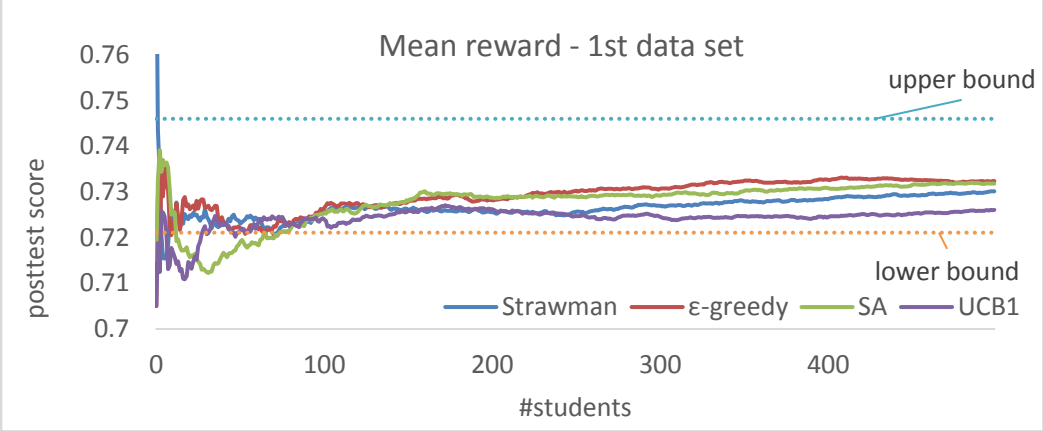


Figure 14. The mean rewards from simulating running the four algorithms on the 1st data set. The dotted lines represent the upper bound (0.746) and lower bound (0.721) in this experiment.

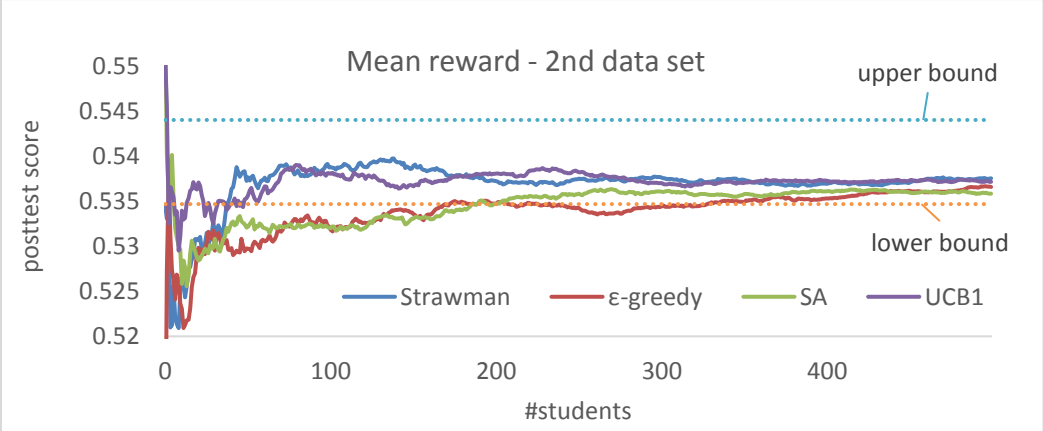


Figure 15. The mean rewards from simulating running the four algorithms on the 2nd data set. The dotted lines represent the upper bound (0.544) and lower bound (0.535) in this experiment.

4.5.3. Comparison of Exploration Rate

With the definition of explorative choice described in Section 4.4.3, the exploration rate at time t is the proportion of 100 runs that is considered as explorative choice. The results of exploration rate are shown in Figure 16 and Figure 17. The first observation from these figures is that every algorithm has higher exploration rate in the second data set, that is because there are more actions in the second data set, which requires the algorithms to explore more to gain information about actions.

Second, the exploration rate differs in the four algorithms. The exploration rate in the ε-greedy algorithm does not change much but stays at very low level, since its probability to explore is $0.1 - \frac{0.1}{|A|}$. While in the Strawman and simulated annealing algorithms, the exploration rate decreases as seeing more and more students, but it drops much faster in the simulated annealing algorithm, which indicates the latter algorithm is more willing to select what it thinks the best. However, according to the fact that they have the closed mean rewards in the two data sets, the simulated algorithm falls into selecting suboptimal action. In the last algorithm, UCB1, the exploration rate stays very high, about 0.3 in the first data set and 0.5 in the second, and it even increases at beginning. It reveals that the UCB1 algorithm still has high probability to explore even it knows the actions well in the long term.

The results for UCB1 surprised us, as the algorithm has optimal asymptotic performance. Two reasons come to mind: first UCB1 is optima--within a constant factor. For proving theorems about computational learning, constant factors can be ignored. For real-world science challenges, they cannot. If UCB1 is within a very large constant factor of the optimal loss for this task, that may be less effective than a technique that does not have a provable loss but does better in real-world experimentation.

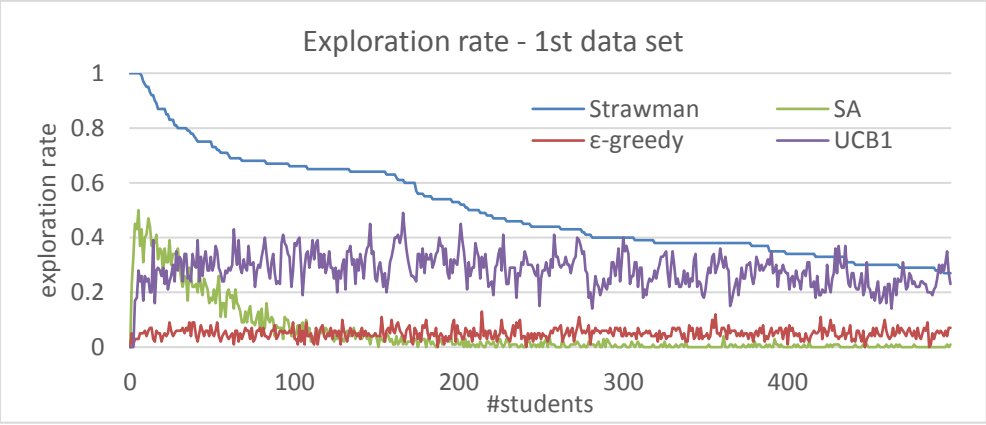


Figure 16. The exploration rate of the four algorithms from the simulation in the first data set.

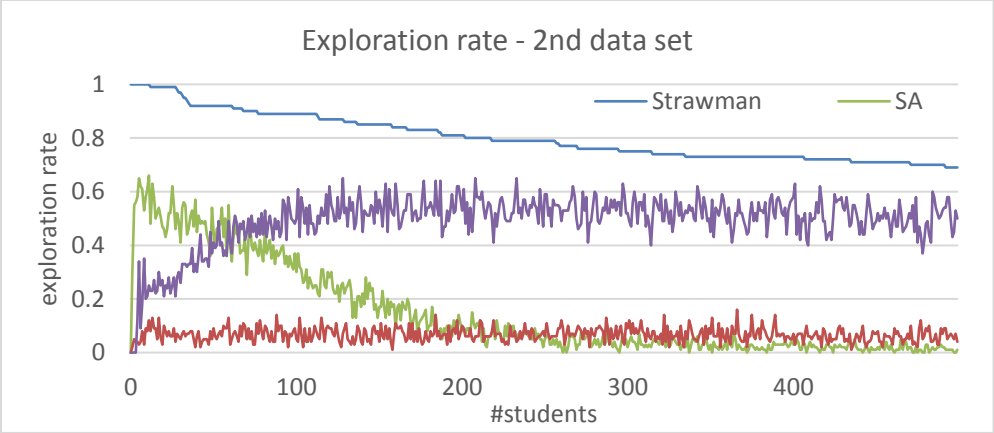


Figure 17. The exploration rate of the four algorithms from the simulation in the second data set.

The second reason is related: UCB1 has a free parameter that can be set by the experimenter. Namely, the base of the logarithm used to weight the number of times this action has been selected. Larger bases in the logarithm give more certainty in the distribution of the rewards across actions and the certainty in the estimate of an action's reward on the basis of its observed reward. We started with a default value of a natural logarithm as other paper reported the formula in using ln rather than log (Auer, Cesa-Bianchi, & Fischer, 2002). However, as the following sections will make clear, the exact value chosen for the logarithm greatly impacts the results.

4.5.4. Algorithms with Different Parameters

UCB1

As discussed in the previous section, UCB1 with natural logarithm is not confident in its estimations based on past observed rewards, and its exploration rate keeps in a high level in the experiments, which causes that it performs worse than other three algorithms in the first data set.

To investigate the effect of logarithm base in UCB1, we experiment the algorithm with three different logarithm bases in the first data set, e, 10, and 100, and the exploration rate after seeing 200 students stays around 0.3, 0.2, and 0.1 respectively. This result is consistent with what we expected, larger logarithm bases bring the UCB1 algorithm more certainty in the estimation of rewards. Moreover, as observed in Figure 18, the UCB1 algorithms with logarithm base 10 and 100 perform slightly better than the algorithm with natural logarithm after seeing 250 students.

Strawman

There is also a parameter that controls the exploration/exploitation tradeoff in Strawman, the significant level in the t-test. Higher significant level results in the algorithm to be more likely to

drop actions, and thus to be more exploitive; while the Strawman with lower significant level is more cautious, and explores more to make drop decisions.

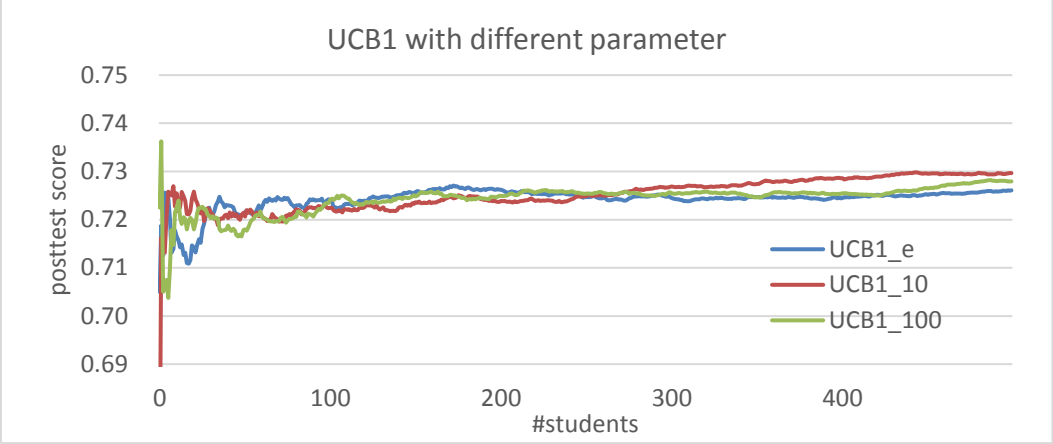


Figure 18. Mean rewards of running the UCB1 algorithm with different logarithm base in the formula, e, 10, and 100, in the 1st data set. The algorithm with lower logarithm base would be more explorative.

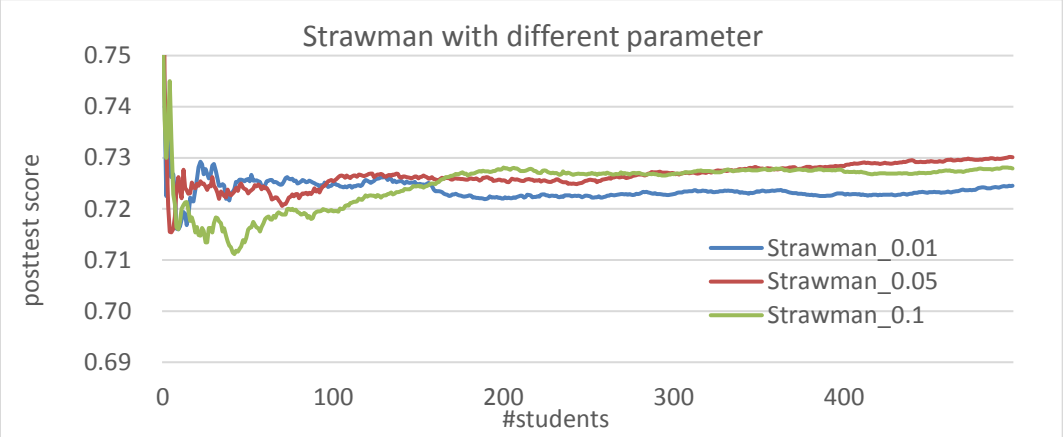


Figure 19. Mean rewards of running the Strawman algorithm with different parameters, 0.01, 0.05, and 0.1, in the 1st data set. The parameter is the significant level in the t-test of comparing pairs of actions. Lower value makes the algorithm to be more explorative.

In this experiment, we run the Strawman with three different significant level, 0.01, 0.05, and 0.1, in the first data set, and the exploration rate is 0.63, 0.3, and 0.2 respectively. As observed in Figure 19, the Strawman with significant level 0.05 outperforms the Strawman with 0.1 after seeing 300 students, because the latter one is more likely to make Type I error – to drop the optimal actions. Moreover, the Strawman with 0.01 has the lowest performance after seeing a couple of students, the reason is that it is still struggling to determine which action is significantly better. But we could image it would have the highest performance in the longer runs.

4.6. Discussion

4.6.1. Performance of Strawman

To put it mildly, we were not expecting Strawman's naïve approach of using t-tests to do very well at this task; therefore, we were surprised at its strong performance. Looking at the data, several possible explanations emerged. First, relative to other bandit problems, educational interventions tend to not very much in their impact. For example, in many board games there are positions where one action will lead to a certain win (maximum possible reward), while another action will lead to a certain defeat (minimum possible reward). Educational interventions, at least those actually tested, tend to have far more moderate impacts. While many interventions we could propose are likely to be harmful, those are unlikely to be proposed by experimenters, let alone survive the IRB process. Second, classic bandit problems are posed in terms of having many levers to select from, and the advantages of clever selection algorithms increases with more potential options. Most human-designed studies have a relatively small number of conditions, thus blunting the benefit of advanced bandit algorithms.

This combination of not having large differences in intervention effectiveness and a relatively small number of interventions affects the performance of bandit algorithms differently. Algorithms such as UCB1 will probably stay high exploration rate, due to the chance of selecting an action being reduced by the times the action has been selected. Relatively smaller variation compared to other bandit tasks means less difference from random actions. In contrast, the approach using t-tests is sensitive to both differences in effectiveness and the confidence in the estimate. Similarly, techniques such as SRT that decreases the temperature as additional data accumulate will do relatively better for this task.

4.6.2. Exploration and Exploitation

A key difference between selection strategies is how they balance the exploration and exploitation in the process of playing actions, and this is measured in this work by exploration rate. A strategy with high exploration rate tends to select actions with equal chance, and thus it has general knowledge about the payoff associated with each action. In the opposite way, the strategy with low exploration rate prefers to stick with selecting the "best" action according to what the strategy has learned currently. However, once it recognizes a suboptimal action as the "best", it is hard to correct the mistake.

A selection strategy exhibits different exploration rate in different environment, and there are two factors might affect the result. The first is the size of action space. With more possible actions, the strategy should explore more to learn the reward distribution for each action. For example, all the

algorithms, introduced in this work, except the ε-greedy, have higher exploration rates at the same time step in the second data set than in the first data set. The second factor is the differences between rewards correspond to the actions. A strategy should explore more to gain enough certainty in estimation of rewards when actions produce similar rewards.

4.7. Conclusion

In this chapter, we analyzed the benefit from replacing random assignment with k-armed bandit algorithms in ASSISTments – students' overall performance could be improved significantly. We also introduce a new selection strategy, Strawman, that is based on statistical t-test, and we compare it with other k-armed bandit algorithms in real ASSISTments experiments. This strategy is more reasonable to be a baseline for the k-armed bandit algorithms than the random selection strategy. To be honest, we were surprised that we have not seen other bandit comparisons using what seems to be the obvious method -- essentially the method taught to all psychology undergraduate students on how to evaluate experiments (with a small caveat that they are not taught to keep testing for effects all the time, the way this method does).

We also introduce a metric, exploration rate, to measure how likely a selection strategy would make an explorative choice at each step. The results reveals that the algorithms exhibit different tradeoff between exploration and exploitation in the process of choosing actions. General speaking, an agent would explore more when it knows little about the whole environment, and thus has high exploration rate. Therefore, the metric reveals, in some aspects, how confident an agent estimates the rewards from choosing possible actions.

Can we do better? Can we make more personalized choice for each student? In the next chapter, I will discuss the effect of contextual bandits in ASSISTments, and I will also investigate different aspect of related features in the educational experiments, such as if a feature has aptitude treatment effect in an experiment, or if a feature has the same effect in multiple experiments.

CHAPTER 5

Contextual Bandits

5.1. Introduction

In normal k-armed bandit problems, we determine which arm to pick based on the history of previous choices and observed rewards. While in contextual bandit problems, we have access to a side information, X, called context. Both bandits share the same goal of picking actions that gives largest total rewards. There are two important issues in contextual bandit problems, how to use the context to estimate rewards, and how to use the estimation to make decisions.

Previous works used various ways to tackle those two issues. For example, Li et al. applied LinUCB (Li, Chu, Langford, & Schapire, 2010), which combined linear regression model with upper confidence bound (UCB), to personalize news articles for each user, with the goal of maximizing the probability of clicking the recommended articles. Lu et al. proposed an algorithm that clustered the context into similar regions and then ran a k-armed bandit algorithm in each region (Lu, Pál, & Pál, 2010). Wang et al. discussed a two-armed bandit problem with different connection between context and reward (Wang, Kulkarni, & Poor, 2005): direct information, the best arm is a function of context, the best arm is not a function of context, and mixed case. Langford and Zhang introduced an ensemble-learning-like contextual bandit algorithm, based on epoch-greedy algorithm, which combined different hypotheses to make decision (Langford & Zhang, 2008). In each epoch, it made one step exploration that selected an arm uniformly at random, and then used the best hypothesis based on the context and historical records to pick the arms in the next exploitation steps (pre-defined in the algorithm).

Recently, bandit problems have attracted much attention in educational area, such as applying k-armed bandits for online optimization of teaching sequences in (Clement, Oudeyer, Roy, & Lopes, 2014), and using contextual bandits framework for personalized learning action selection that aims to maximize students' success on the follow-up assessment in (Lan & Baraniuk, 2016). However, very few works use contextual bandits in the educational area. In this dissertation work, I focus on investigating the effect of contextual bandits in real ASSISTments experiments.

In the rest of this chapter, I will illustrate why we need contextual bandits in ASSISTments, and then I will answer the three research questions related to this problem: 1). What context should be incorporated into the bandits? 2). How to organize the context? 3). How much benefit?

5.2. Context Makes Better Personalization

Bandit algorithms are able to make use of context in order to make better decisions. For example, perhaps a certain intervention works well for highly-motivated students, but is ineffective for unmotivated students. It would be useful if we were able to provide our bandit algorithm with some context about the learner to improve its decision making. Therefore, we set out to capture that bit of context to potentially aid decision making. Take the second data set in Section 4.5.1 as an example. Students' performance in each condition is shown in Table 12, according to this table, the best performance we can achieve is 0.544 by assigning all students into "With image".

To illustrate the effect of contextual bandits, we disaggregate the original data sets with a student feature, called "performance in previous 3 days". This feature represents the percent of correctness of all items a student practiced in ASSISTments in the three days just before the experiment. For example, suppose a student starts the ASSISTments experiment at time t_s, then we search all items from the student's records in the system with the corresponding time, t, having: $0\ day < t_s - t \leq 3\ days$.

We separate the data set into three groups according to the feature x, proportion of correctness over previous 3 days: high ($0.85 \leq x \leq 1$), moderate ($0.7 \leq x < 0.85$), and low ($0 \leq x < 0.7$). We chose those cutpoints to split the students into roughly 3 equal-sized groups. Students' performance in each group of the first data set is in Table 13. By picking the optimal condition for each group, "No image" for students in low and high level, and "With image" for students in moderate level, the upper bound is: $(0.562 * 75 + 0.54 * 66 + 0.586 * 65)/206 = 0.563$, which is higher than the one without being disaggregated by this feature. However, this could be resulted from overfitting or cherry-picking, the feature that is useful here might not be effective in the other data sets. Therefore, a feature should be evaluated across different experiments.

In summary, proper context brings aptitude treatment effect in students' learning process, and thus results in better upper bound for the bandit algorithms. But which context could bring such effect? Moreover, we disaggregated the data set with only one feature in this experiment, how to make the disaggregation with multiple features, in order to obtain the best aptitude treatment effect? Finally, we just computed the estimated upper bound of applying k-armed bandits on the data set, what are the real results of running the bandits without context and the contextual bandits on the data set? Could context still bring better results? These questions will be answered in the next sections one by one.

Table 12. Students' posttest scores in each conditional problem set in the original data set. Students with the bolded condition have higher mean posttest.

Condition	#Students	Mean	Std.
Correctness only	74	0.518	0.214
With image	**63**	**0.544**	**0.229**
No image	69	0.542	0.201

Table 13. Students' posttest scores in the data set with disaggregated by students' performance in previous 3 days.

Performance	Condition	#Students	Mean	Std.
Low	Correctness only	29	0.473	0.221
	With image	25	0.544	0.227
	No image	**21**	**0.562**	**0.203**
Moderate	Correctness only	22	0.513	0.209
	With image	**18**	**0.54**	**0.263**
	No image	26	0.489	0.218
High	Correctness only	23	0.58	0.206
	With image	20	0.547	0.21
	No image	**22**	**0.586**	**0.172**

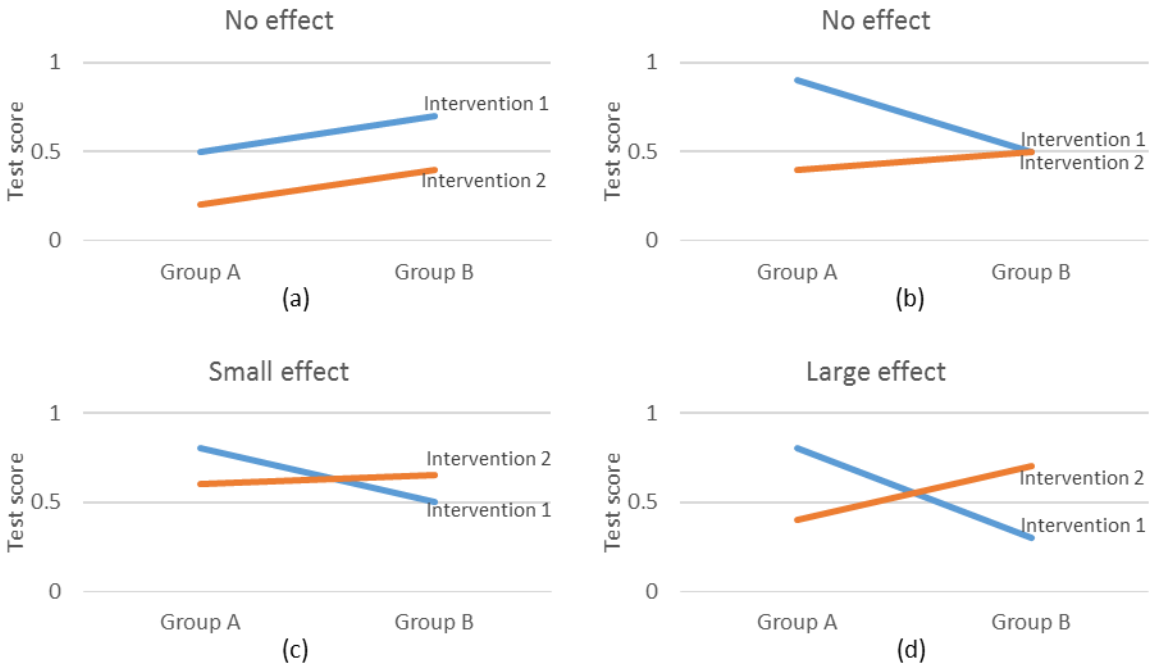


Figure 20. Different effectiveness of interventions in two groups of students. In the cases (a) and (b), the interventions have no effect because the optimal intervention is intervention 1 for both groups. In the cases (c) and (d), the interventions have treatment effect in the groups, intervention 1 is optimal for the group A, while intervention 2 for group B. According to the difference between interventions in each group, the interventions have small effect in the case (c), and large effect in the case (d).

5.3. Feature Evaluation

5.3.1. Mechanism

1. Cross effect

According to work in (Pashler, McDaniel, Rohrer, & Bjork, 2008), a learning-style hypothesis is accepted if the optimal learning method of one kind of learner is different from the optimal learning method of the other kind of learner, and two crossed lines are shown in the plot of learning outcomes with respect to learning methods and learner styles, as illustrated in Figure 20. In this work, we also regard a feature is useful if and only if the feature has such "cross effect": the optimal option differs in the groups which are disaggregated by the feature.

We define the cross effect as the improvement of target value from picking the overall best action to picking the best action for each group. If there is no cross effect, then there is no difference with disaggregated by the feature, and thus the improvement is 0. For example, in Figure 20, there is no cross effect in the cases (a) and (b), and positive cross effect in (c) and (d). The metric to evaluate a feature, x, in a data set, D, is defined as:

$$Gain(x, D) = \sum_{g_i \in G} \frac{mean\left(D_{g_i, a=\hat{a}\left(D_{g_i}\right)}\right) * |D_{g_i}|}{|D|} - mean(D_{a=\hat{a}(D)})$$

$$\hat{a}(D) = argmax_{a_j \in A}\left(mean\left(D_{a=a_j}\right)\right)$$

Where G is the set of all groups disaggregated by the feature, D_{g_i} is the data in the group g_i, A is the set of all possible actions, $D_{a=a_j}$ is the instances in D of which action is a_j, $mean(D)$ is the mean reward of all instances in D, and $\hat{a}(D)$ represents the best action in the data set D. Taking the data set in 5.2 as an example, the feature "student's performance in previous 3 days" separates the original data set into three groups, as shown in Table 12 and Table 13, and the corresponding cross effect is: $(0.562 * 75 + 0.54 * 66 + 0.586 * 65)/206 - 0.544 = 0.019$.

In the rest of this dissertation work, we only consider binary disaggregation. If a feature x is continuous, given a split value x_i, it separates the data into two groups, one with the feature value $x \leq x_i$, and the other with $x > x_i$; if x is discrete, given one of its possible value x_i, it separates the data into two groups, one with $x = x_i$ and the other with $x \neq x_i$.

2. Factorial Analysis of Variance (ANOVA)

Factorial ANOVA measures whether a combination of independent variables predict the value of a dependent variable (Devore, Farnum, & Doi, 2013). In this work, we consider the feature to be evaluated as one independent variable, the tutorial condition in ASSISTments as the other independent variable, and the reward as dependent variable. P-value outputted from factorial ANOVA is used to measure significance of the combination.

To evaluate a feature in an experiment, we will use its every possible value as a cut point, and compute the corresponding cross effect and p-value, output from factorial ANOVA. If there are positive cross effect in at least one splits, then we consider the feature is useful in that experiment. With a pre-defined p-value threshold, we can screen out the features that cannot bring a significant cross effect.

5.3.2. Experiments

Data

This data set is collected from 22 ASSISTments experiments (Heffernan & Heffernan, 2014). In each experiment, a student is randomly assign into one of two groups, and the two groups are associated with different tutoring conditions, such as video feedback and text feedback. And each experiment has a unique pair of conditions. After removing the missing values, the experiment with minimum size has 121 students, 1640 for the maximum, and there are 10690 students in total. In the rest of this dissertation work, the experiments of contextual bandits are executed on this data set.

Features and Reward Function

In each experiment, students are learning a specific skill by practicing related problems, and they are required to obtain 3 correct in a row to complete the experiment. In this dissertation work, we will investigate which condition is optimal for each student, we defined the reward function as:

$$reward = \begin{cases} if\ complete: (3/\#practices)^{0.7} \\ otherwise: \ 0 \end{cases}$$

Where "#practices", also called mastery speed, is the number of problems a student takes to complete the experiment. The range of reward value is from 0 to 1, and the larger the better. Thus, there are exactly two different conditions in each experiment. In this work, we will evaluate 8 features in the experiments:

1. Prior completion rate: student's completion rate in all prior participated experiments;
2. Prior percent of correctness: student's percent of correctness in all prior practiced problems;

3. Prior mastery speed: student's average mastery speed in all prior participated experiments, and computed as z-score over all students;
4. Imputed gender: student's gender that is imputed according to the student's name. This feature has three values, female, male, and unknown;
5. Location: this is where student live, categorized as urban, suburban and rural;
6. Prerequisite performance: student's percent of correctness in the problems of which related skills are prerequisite to the skill he/she is learning;
7. Percent of correctness in previous 3 days: student's percent of correctness in the problems that were practiced within 3 days;
8. Learning rate in previous 3 days: the average learning rate in the experiments a student participated within 3 days. Learning rate represents how fast a student learns a skill. It first computes the percent of correctness in each item, and then construct a linear regression model on those values with respect to number of practices items. The slope in the linear regression model is considered as student's learning rate in this skill. For example, if a student's responses in a skill are: wrong, wrong, wrong, right, and right, then the series of percent of correctness is: 0, 0, 0, 25%, and 40%. And the corresponding linear regression model is shown in Figure 21, thus, his learning rate is 0.105.

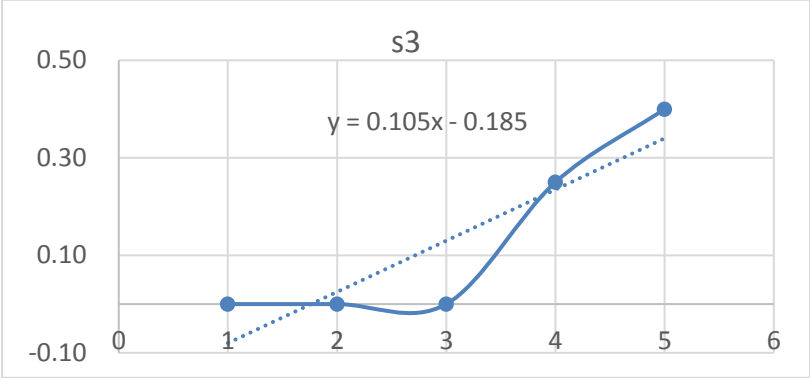


Figure 21. A linear regression model constructed on a student's %correctness with respect to number of practiced items. In this sample, the student's responses is wrong, wrong, wrong, right, and right, so the series of %correctness is 0, 0, 0, 25%, and 40%. The slope in the model, 0.105, is this student's learning rate.

Results

1. Location

In the data set, 10% of students live in Urban settings, 13% in rural, and 77% in suburban. In each experiment, the location values are almost the same, since most of students who participated in the experiment are from the same school. Therefore, personalizing based on this feature could not help

many students, and the corresponding cross effect would be very small. For those reasons, I will disregard this feature.

2. Imputed gender

To compute the cross effect of imputed gender for a data set, first, we use each value to separate the data set into two groups, for example, "female" and "not female". Then we compute the cross effect on the separated groups. Finally, we pick the best one as the cross effect of imputed gender in the data set, and the corresponding p-value for the best cross effect is computed.

In the 15 out of 22 ASSISTments experimental data sets, imputed gender results in a positive cross effect, which means the feature is useful in personalization in those experiments. Of those positive cross effects, only 1 with $p \leq 0.05$, 2 with $p \leq 0.1$, 5 with $p \leq 0.2$, and 9 with $p \leq 0.3$. So most of the effects are not statistically reliable.

3. Prerequisite performance

Because students did not learn the prerequisite skills, or teachers thought those skills were not important, or other reasons, we obtained a very sparse prerequisite performance in the data sets. As shown in Table 14, there are only 5 experimental data sets containing at least one student who started learning the related prerequisite skills. In the other 17 experiments, no one started learning the pre-required skills. Therefore, we will consider using this feature in the first two experiments, so as not to lose too much data.

To evaluate prerequisite performance in a data set, we will consider every value in $\{x_1, x_2, \dots x_{100}\}$ as a cut point, where x_1 is the lowest value in the data set, $x_{i+1} = x_i + stepsize$, and $stepsize = (hv - lv)/100$ (hv – highest value in the data set, lv – lowest value in the data set). For a given value, say x_i, which separates the data set into two groups, $x \leq x_i$ and $x > x_i$, and then compute the cross effect and p-value on the separated groups. The computed cross effect and p-value with respect to cut point are shown in Figure 22 and Figure 23 respectively. In both data sets, prerequisite performance results in positive cross effect, and the result is more reliable when cut point is less than 0.3 in the data set 293151, while the result is more reliable when cut point is more than 0.5 in the data set 226210.

Table 14. The proportion of data having prerequisite performance in each experiment.

Experiment	Has prerequisite performance available
293151	88%
226210	80%
243393	50%
263109	40%
303899	2%

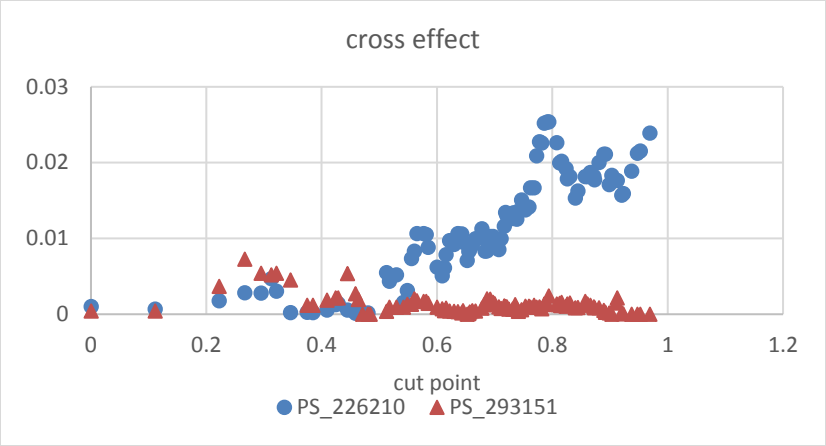


Figure 22. The cross effect of prerequisite performance in two data sets.

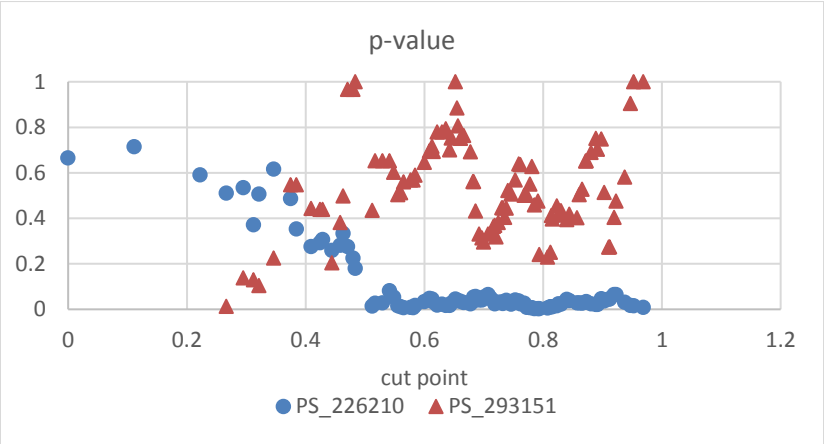


Figure 23. The p-value of prerequisite performance in two data sets.

4. Other five features

For each other feature, since they are all continuous features, similar to as prerequisite performance, they are also evaluated with 100 possible cut point in each experiment. The results show that those features are useful at least one of in the 22 experiments, because each of them can produce positive cross effect in the experiments. To investigate the generalizability of a feature, for each cut point, we count in what proportion of the 22 experiments, this feature produces the cross effect, and the corresponding p-value with $p \leq 0.05$, $p \leq 0.1$, $p \leq 0.2$, and $p \leq 0.3$. To reduce the effect of noise, we smooth the result by replacing the cross effect of each value with the average of its 5 neighbors, including itself. The results of those features are shown in Figure 24 to Figure 28 respectively.

As shown in the figures, the effectiveness of prior percent of completion and percent of correctness in 3 previous days does not vary too much with respect to their cut points; while the feature, prior mastery speed, is more effective in the experiments between -1.0 and 1.0, and the feature, learning

rate in 3 previous days are more effective between -0.1 and 0.1; in general, the effectiveness of prior percent of correctness increases with cut point, but it drops afterwards, and the largest proportion is approximately 83% when the cut point is 0.72.

5.3.3. Conclusion

In summary, we analyzed the effect of 7 features in making better personalized treatment in the 22 ASSISTments experiments. The results show that those features, except prerequisite performance, are useful in all 22 experiments. There are only two experiments that have enough students with prerequisite performance, but this feature can significantly improve students' learning outcomes in the two experiments. Therefore, in the next step, modeling multiple features together, I will use prerequisite performance in the experiments, 293151 and 226210, and the other 6 features in all 22 experiments.

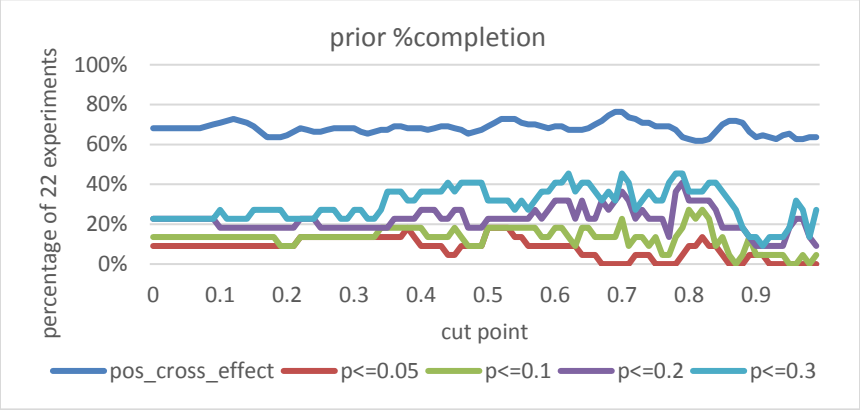


Figure 24. The percentage of positive cross effect of prior %completion and statistically reliable effect in the 22 experiments.

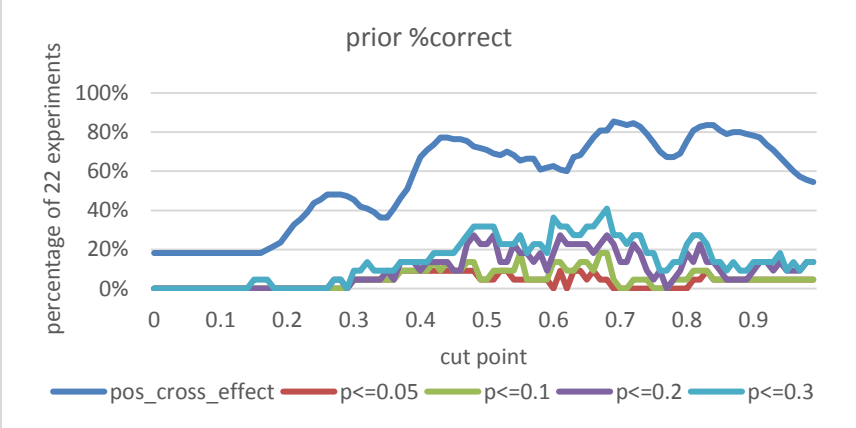


Figure 25. The percentage of positive cross effect of prior %correct and statistically reliable effect in the 22 experiments.

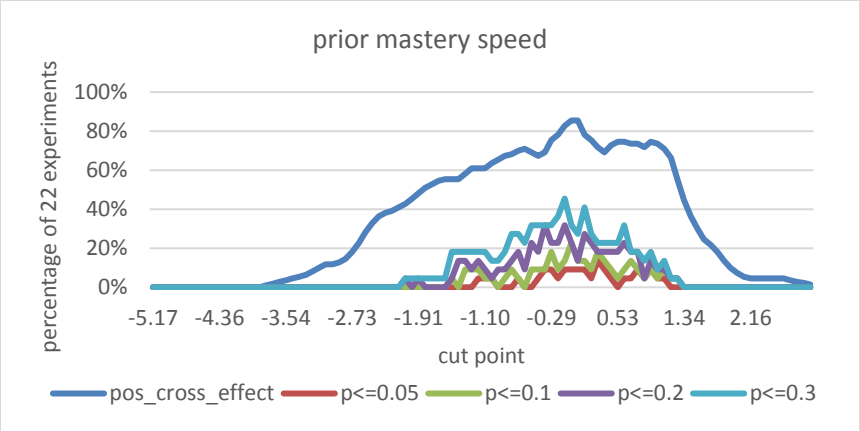


Figure 26. The percentage of positive cross effect of prior mastery speed and statistically reliable effect in the 22 experiments.

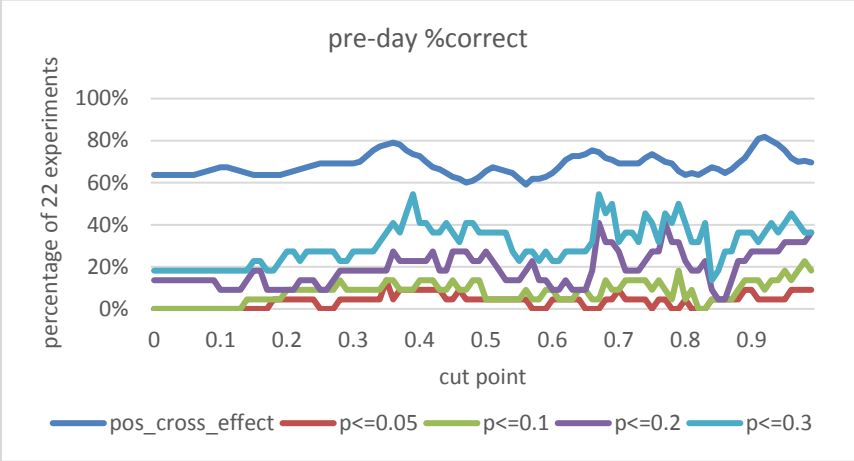


Figure 27. The percentage of positive cross effect of %correctness in 3 previous days and statistically reliable effect in the 22 experiments.

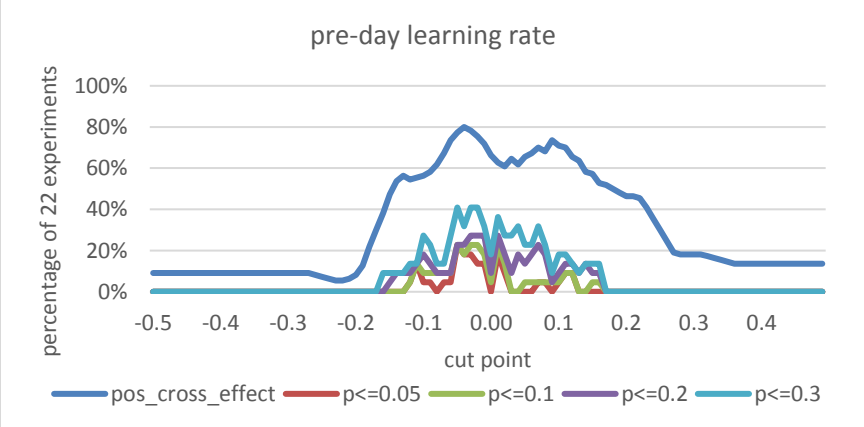


Figure 28. The percentage of positive cross effect of learning rate in 3 previous days and the statistically reliable effect in the 22 experiments.

5.4. Modeling Multiple Features

5.4.1. Introduction

The ability to customize instruction to individuals is a great potential for adaptive educational software. Unfortunately, beyond mastery learning and learner control, there has not been much work with adapting instruction to individuals. In this work, we focused on constructing a decision tree that produced treatment effect for different groups of students, and thus to make customization for each student.

There are many research works that utilize student's feature to personalize. Even since Bloom's paper showing a 2 standard deviation effect size of individual tutoring (Bloom, 1984), the computer's ability to adapt instruction has been pitched as a solution. However, most educational software does little adaption to the individual beyond mastery learning and learner control. Each student (largely) sees the same help messages and the same instruction. Mastery learning enables students to keep practicing a skill until they have mastered it. Learners maintain a degree of control, such as selecting which story to read next or which type of help to receive (e.g., (Mostow & Beck, 2006)). In another example, Dagger et al. introduces a e-learning system which composes adaptive courses according to the concept space, the pedagogical strategy, the learning activities, and the adaptive mechanisms, as well as user's status and preferences (Dagger, Wade, & Conlan, 2005).

We define "option" as the choices a tutoring system has for teaching at a particular moment. For example, the system could have the options of showing a video on Pythagorean theorem, or going back on working on a prerequisite skill. The reason to implement customization service is that students' learning outcomes might differ in tutoring options, because they have different learning styles (Cha, Kim et al., 2006; Pashler, McDaniel, Rohrer, & Bjork, 2008), prior knowledge (Botelho, Wan, & Heffernan, 2015; Wan & Beck, 2015), or other factors that affect which type of instruction is most effective for *this* learner. Therefore, estimating the student's potential outcomes for each option is the key for customization.

In this dissertation work, we explore identifying the groups of students such that the interventions have treatments effect in these groups. We use decision trees to estimate students' learning outcomes, and thus to make customization. There are four reasons to use the decision tree technique. First, it implicitly performs feature selection as at each step it selects the feature that provides maximum information. Second, it is easy to either manually or automatically perform a rule extraction from a tree. Third, as a consequent of the first two points, a decision tree's structure is easy to understand and explain to human practitioners. If teachers have a say in what software is

used in their classrooms, using a technique that is (relatively) easy to explain is essential. Fourth, it is easy to combine decision tree structure with k-armed bandit algorithms. As discussed in previous sections, it is very hard and time consuming to apply a k-armed bandit algorithm for each combination values of multiple features. In this dissertation work, we will apply a bandit algorithm for each leaf node in the decision tree, while we can control the complexity of the decision tree by setting with appropriate parameters.

Unlike traditional classification problem, such as in (Cha, Kim et al., 2006), where every student is marked with which option is optimal for him/her, we focus the problem where we do not know *a priori* how each student best learns. Even worse, we only know how a student would perform with one of the possible options. Therefore, we must find commonalities in what types of students learn better from one intervention vs. another. Consequently, we need to employ the decision tree to determine difference of outcome between options for each student, based on the corresponding features, and then output the best option for this student.

One possible approach to solve the problem is to build a model to estimate the effect of each possible option, and then select the one with the best estimated outcome for this student (Cha, Kim et al., 2006; Kim, Lee, Shaw, Chang, & Nelson, 2001). However, it has three disadvantages.

1. The mechanism would be very complicated if the size of possible options is large, like in course recommendation systems (Weber, Kuhl, & Weibelzahl, 2001; Williams, Li et al., 2014), each course could be considered as an option, and there might be tens of different options.
2. It would be hard to extract rules from the decision trees if they are constructed on different set of features or nodes are split with different values.
3. The constructed models might be meaningless, and learn about what is termed "unacceptable evidence" (Pashler, McDaniel, Rohrer, & Bjork, 2008) of a meaningful interaction. For example, Figure 29 shows that students, no matter with high or low knowledge level, would always perform better with option 1. This conclusion does not need a decision tree. The reason for the overly complex model is that traditional decision tree algorithms use criteria such as information gain, Gini index, impurity, and so on, to construct a tree with the least error in prediction (Blockeel & De Raedt, 1998; Breiman, Friedman, Stone, & Olshen, 1984; Quinlan, 2014). However, in this problem, we should focus on building decision trees to find splits that would alter which option works best. As a result, the decision tree is "acceptable evidence", i.e., the best option differs for different types of students.

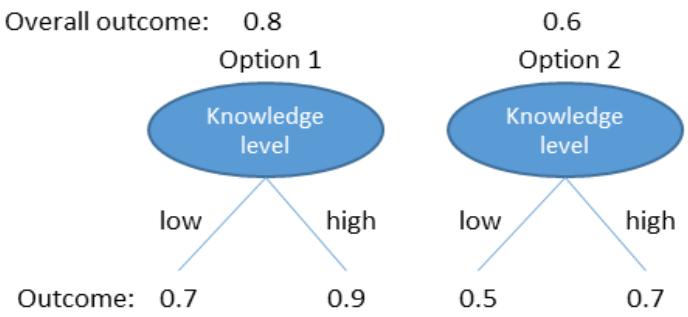


Figure 29. Example of decision trees based on Pashler et al.'s framework of "unacceptable evidence" (Pashler, McDaniel, Rohrer, & Bjork, 2008). These two decision trees are useless, because students would always have a better outcome with option 1.

To overcome those disadvantages, we will introduce a new decision tree algorithm in this work that constructs only one decision tree for all possible tutoring options. In this manner, we address our research question: which option is optimal for a particular type of student? Moreover, many studies focus on investigating the effect of **an** intervention. However, this work explores a set of 22 experiments to find a mechanism for customizing instruction to an individual student.

5.4.2. Background

Decision trees are a machine learning method for constructing prediction models. Starting at the whole data space, it selects an appropriate feature, according to a certain criterion, to split the data space into several sub-groups, and each sub-group is recursively deployed the split process. The tree keeps growing until stopping criteria are met, such as the maximum tree depth have reached, or all instances in the data space belong to a single value of target variable, or some rules else (Blockeel & De Raedt, 1998). As a result, each internal node represents a feature that splits the tree, and each leaf node is marked with a target value or a probability distribution over the target values.

Common criteria that are used in split when the target variable is discrete include information gain (Quinlan, 1987), gain ratio (Quinlan, 2014), Gini index (Breiman, Friedman, Stone, & Olshen, 1984), distance measure (De Mántaras, 1991), and twoing criteria (Breiman, Friedman, Stone, & Olshen, 1984). Taking Gini index as an example to illustrate the split process.

Gini index is defined as: $Gini(D) = \sum_{y_i \in Y} \frac{|D_{y=y_i}|}{|D|}$, where D is current data space to be split, Y is the set of possible target values, and $D_{y=y_i}$ is the data set where every instance has the target value y_i. The Gini gain, which is used to evaluate the features at each split step, is defined as:

$$Gini\ Gain(x, D) = Gini(D) - \sum_{x_i \in X} \frac{|D_{x=x_i}|}{|D|} * Gini(D_{x=x_i})$$

Where x is the feature to be evaluated, and X is the set of its possible values. The algorithm tends to pick the feature with the largest Gini gain in the split process.

If the target variable is continuous, the tree is referred as regression tree, and each leaf node represents the mean target value of all training instances that follow into this node. A common used criterion in the split process is sum of squared error (SSE):

$$SSE(D) = \sum_{d \in D} (y_d - \bar{y})^2$$

Where $\bar{y}$ is the mean of target variable in the data space D. And the criterion in selecting feature to split is:

$$gain(x, D) = SSE(D) - \sum_{x_i \in X} SSE\left(D_{x=x_i}\right)$$

Another important step in decision tree induction is discretization, this is deployed on the continuous features. A simple discretization technique is bin method, which uses a continuous feature to separate the data into groups with equal-width or equal frequency. Other sophisticated methods, as summarized in (Liu, Hussain, Tan, & Dash, 2002), are categorized as entropy -based methods, dependency-based methods, accuracy-based methods, and merging methods.

In this work, our discretization method makes a binary split, like in ID3 (Quinlan, 1986) and C4.5 (Quinlan, 2014), and it tends to find a cut point that outputs two groups with the most "discriminability" on the options. More details are discussed in the next section.

5.4.3. Methodology

Split Criterion

Here we also use cross effect, as defined in Section 5.3, as split criterion to evaluate features in decision tree construction. The metric is defined as:

$$Gain(x, D) = \sum_{g_i \in G} \frac{mean\left(D_{g_i, a=\hat{a}\left(D_{g_i}\right)}\right) * |D_{g_i}|}{|D|} - mean(D_{a=\hat{a}(D)})$$

$$\hat{a}(D) = argmax_{a_j \in A}\left(mean\left(D_{a=a_j}\right)\right)$$

Where G is the set of all groups disaggregated by the feature, D_{g_i} is the data in the group g_i, A is the set of all possible actions, $D_{a=a_j}$ is the instances in D of which action is a_j, $mean(D)$ is the mean reward of all instances in D, and $\hat{a}(D)$ represents the best action in the data set D.

Taking the data in Table 15 as an example to illustrate how to compute the cross effect. Overall speaking, the option 0 is better than the option 1, since the mean target value of all students with

option 0 is 0.74, larger than the mean target value of option 1, 0.72. Suppose the cut point for the feature A is 0.5, then the data is split into two groups: group 1 – "$A \leq 0.5$" and group 2 – "$A > 0.5$". In the group 1, there are two students with option 0, the 6th and 9th student, and their mean target value is (0.9+0.9)/2=0.9. Consequently, the mean target value for the students in group 1 with option 1 is 0.63, group 2 with option 0 is 0.63, and group 2 with option 1 is 0.85. By picking the optimal option for each group, which is option 0 for group 1 and option 1 for group 2, the cross effect is: $\frac{0.9*|D_{group\,1}|+0.85*|D_{group\,2}|}{|D|} - 0.74 = \frac{0.9*5+0.85*5}{10} - 0.74 = 0.135$.

Discretization

In this work, we deploy binary split in decision tree induction. To evaluate a discrete feature x, for each possible value x_i, we consider the data space is divided into two groups, the one with "$x = x_i$" and the other with "$x \neq x_i$", and then we compute cross effect according to this division. Finally, the best one is marked as the cross effect of x.

To use a continuous feature to split the decision tree, we need to set a cut point. To discretize a continuous feature f, as its values are denoted in order as $\{x_1, x_2, \dots, x_m\}$, each value will be considered as a cut point, so that a value, x_i, will divide the data into two groups, one containing the instances with $x \leq x_i$, and the other with $x > x_i$. The cross effect according to this division will be computed. Therefore, we need to examine $m - 1$ possible values. Figure 30.a shows all computed cross effect of the feature A in Table 15.

Table 15. A sample data set, which contains a continuous feature, A, a discrete feature, B, an option variable with possible value 0 and 1, and a continuous target variable.

student	A	B	Option	Target
1	0.4	0	1	0.7
2	0.8	0	0	0.6
3	0.7	1	0	0.8
4	0.1	0	1	0.6
5	0.2	1	1	0.6
6	0.4	0	0	0.9
7	0.6	0	0	0.5
8	0.9	1	1	0.8
9	0.2	1	0	0.9
10	0.8	1	1	0.9

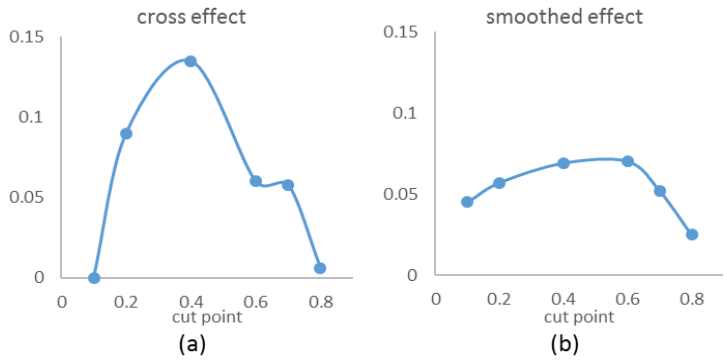


Figure 30. (a) the cross effect with respect to each value of feature A in Table 15; (b) the smoothed effect calculated by taking the average cross effect of 5 neighbors for each cut point.

After computing the cross effect for all possible values of a continuous feature, we are not going to pick the value with the best cross effect as the cut point for this feature. The reason is that there might be noise in the data, which is very possible in real experimental data. And the noise would cause high-frequency oscillations, so the best cross effect might be obtained by chance. To reduce the effect of noise, we smooth the result by replacing the cross effect of each value with the average of its 5 neighbors, including itself. That is, for each feature value, f_i, with the corresponding cross effect ce_i, we will compute a smoothed effect, $se_i = (ce_{i-2} + ce_{i-1} + ce_i + ce_{i+1} + ce_{i+2})/5$. Figure 30.b is the smoothed effect of feature A in Table 15.

Finally, we pick the value with the best smoothed effect as the cut point for a continuous feature. We also use factorial ANOVA(Devore, Farnum, & Doi, 2013) on the resulted disaggregation to compute p-value for both continuous feature and discrete feature. This p-value will be used in decision tree induction.

Decision Tree Induction

Our method is a top-down induction method, starting with the whole data set, it keeps splitting the data set into two sub data sets, until every possible split meet stop criterion 1 or 2:

Stop criterion 1: in one of the two sub data sets, there is an option, such that the number of instances with the option is less than n, a parameter of minimum size used in the induction.

Stop criterion 2: the resulted p-value $> l$, where l is another parameter in the induction, which indicates the significant level of splitting.

The split procedure is shown in Figure 33, line 1-3 is initialization; line 4-12 is used to pick the feature alone with its corresponding cut point that produces the most significant split. A significant split means the p-value, computed from apply factorial ANOVA on the split groups, is less than or

equal to a pre-defined significant level l; line 13-18 constructs an internal node with the feature and cut point, and then recursively runs the split procedure on the two split sub data sets; if such feature does not exist, which means the stop criterion 2 has been reached, then the tree stop growing, so line 20 creates a leaf node. Finally, this procedure outputs either an internal node or leaf node.

Another two important procedures of our method, the procedure of picking the cut point with the best valid partition and the procedure of dividing a data set into two sub sets, are shown in Figure 31 and Figure 32 respectively. The former procedure takes a data set, a feature and minimum size as input, and it examines all valid partitions that do not meet the stop criterion 1. Thus, it outputs the best smoothed effect and the associated cut point. Input of the later procedure includes a data set, a feature, and a value of that feature. The procedure divides the input data set into two sub data sets, according to whether the feature is discrete or not, and it outputs the two sub data sets.

```
PROCEDURE pickBestCutpoint(D, f, n):
1:  best_cut ← −∞
2:  best_effect ← −∞
3:  FOR each possible value f_i of f:
4:      (D_1, D_2) ← divide(D, f, f_i)
5:      IF meet stop criterion 1:
6:          CONTINUE
7:      END IF
8:      se ← computeSmoothedEffect(D, f, f_i)
9:      IF se > best_effect:
10:         best_effect ← se
11:         best_cut ← f_i
12:     END IF
13: RETURN (best_cut, best_effect)
```

Figure 31. The process of picking the cut point for a given feature, $\boldsymbol{f}$, in a data set, $\boldsymbol{D}$, that produces the best smoothed effect, with a parameter, $\boldsymbol{n}$, that is used in the stop criterion 1, according to the method described in Section "Discretization".

```
PROCEDURE divide(D, f, f_i):
1:  IF f is discrete:
2:      D_1 ← D_{f=f_i}
3:      D_2 ← D_{f≠f_i}
4:  ELSE
5:      D_1 ← D_{f≤f_i}
6:      D_2 ← D_{f>f_i}
7:  END IF
8:  RETURN (D_1, D_2)
```

Figure 32. The process of dividing a data set into two sub sets, given a feature and a value.

```
PROCEDURE split(D, R, n, l):
1:  best_feature ← ''
2:  cut ← −∞
3:  best_effect ← −∞
4:  FOR each feature f in R:
5:      (f_i, e) ← pickBestCutpoint(D, f, n)
6:      p ← computePValue(D, f, f_i)
7:      IF p ≤ l AND e > best_effect:
8:          best_effect ← e
9:          best_feature ← f
10:         cut ← f_i
11:     END IF
12: END FOR
13: IF best_feature ≠ '':
14:     root ← createInternalNode(D,best_feature, cut)
15:     R ← R − f
16:     (D_1, D_2) ← divide(D,best_feature, cut)
17:     root.left = split(D_1, R, n, l)
18:     root.right = split(D_2, R, n, l)
19: ELSE
20:     root ← createLeafNode(D)
21: END IF
22: RETURN root
```

Figure 33. The split process in decision tree induction. It takes current data set, a set of features, and other two parameters that are used in the stop criteria as input, and it outputs the root node of the tree constructed based on the input data set.

5.4.4. Experiments and Results

Experiment Setup

We use 5-fold cross validation to evaluate our method on two types of data sets, one is the simulated data set that is generated with pre-defined distributions, and the other is collected from 22 ASSISTments experiments.

The process of evaluating trained model in this work is different from in the traditional classification problems, since we going to evaluate how well students would have done by given the customized options, not how well the model predicts. After a model is trained by the training set, each separated group, according to the model, is marked with an option which brings the best mean target values. For example, in the decision tree constructed by our method, each leaf node can be considered as a group.

To evaluate the trained model in the testing set, first, the testing set is also assigned into corresponding groups based on the model structure. And then for each group, the mean target value of the instances in the testing group with the marked option is the estimated value. Finally, how well a model is making customization is computed by taking the average target value of all groups.

Simulated Data

1. Data Generation

This simulated data set contains 6 features, 1 option feature with 2 possible values (0 and 1), and 1 continuous target variable. The 6 features are generated with uniform distribution, and the target variable with normal distribution. Parameters of the distributions for first two features, f1 and f2, and the target variable are defined in the Table 16. For example, in the group 1, f1 is generated with a uniform distribution $U(0,0.5)$, and f2 with $U(0,0.8)$; the target value with the option 0 is generated with a normal distribution $N(0.5,0.2)$, the target value with option 1 is generated with $N(0.4,0.2)$. The other four parameters are generated with $U(0,1)$ for all groups. We generated 200 instances for each group, 100 with option 0 and 100 with option 1. Therefore, according to these distributions, the optimal option for group 1 and group 4 is option 0, while option 1 for group 2 and group 3.

2. Results

The mean target value of the generated data in each group is shown in Table 17. The best customization method is to assign each user with the right option, so the upper bound of this data set that the best method can achieve is: $(0.482 + 0.799 + 0.633 + 0.918)/4 = 0.708$.

We use the method described in previous section to construct decision trees on this simulated data with different significant levels, 0.05, 0.1, 0.2, 0.3, 0.5, and 0.9, the other parameter, minimum size, is set to be 20 in all trees. We also compared the constructed decision trees with two methods, random selection and always pick the option that has the best overall mean target value in the training set. As shown in Figure 34, the results of decision trees are very closed to the upper bound, and they are much better than random selection and method of picking the best. Moreover, the decision tree with parameter 0.05 is better than the other models, and it is significantly ($\alpha < 0.001$) better than the method of picking the overall best, the reason could be that it is less overfitting to the training set.

Table 16. Parameters of the distributions used to generate simulated data set.

			option 0		option 1	
	f1	f2	μ	σ	μ	σ
group 1	≤0.5	≤0.8	0.5	0.2	0.4	0.2
group 2		>0.8	0.3	0.2	0.8	0.2
group 3	>0.5	≤0.8	0.4	0.2	0.6	0.2
group 4		>0.8	0.9	0.2	0.2	0.2

Table 17. The mean target values of the generated data.

	option 0	option 1
group 1	0.482	0.371
group 2	0.278	0.799
group 3	0.443	0.633
group 4	0.918	0.208

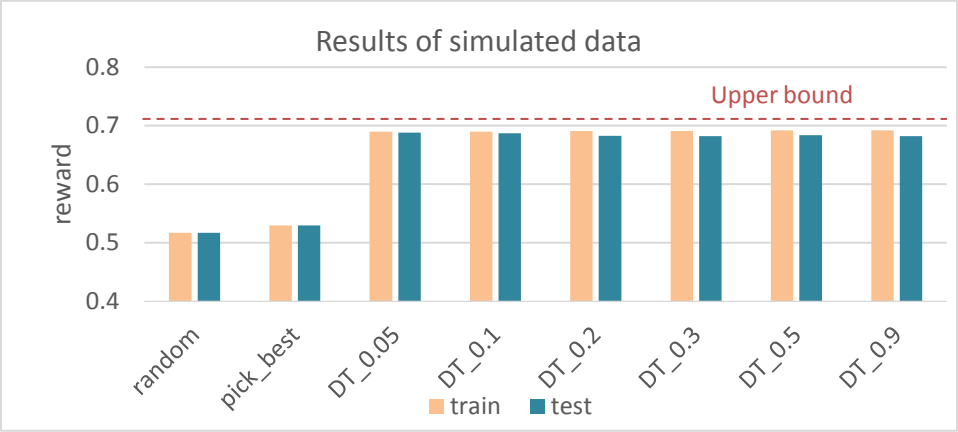


Figure 34. Results of running our decision tree algorithm with different significant levels on the simulated data, compared with random selection method and the method of picking the overall best option.

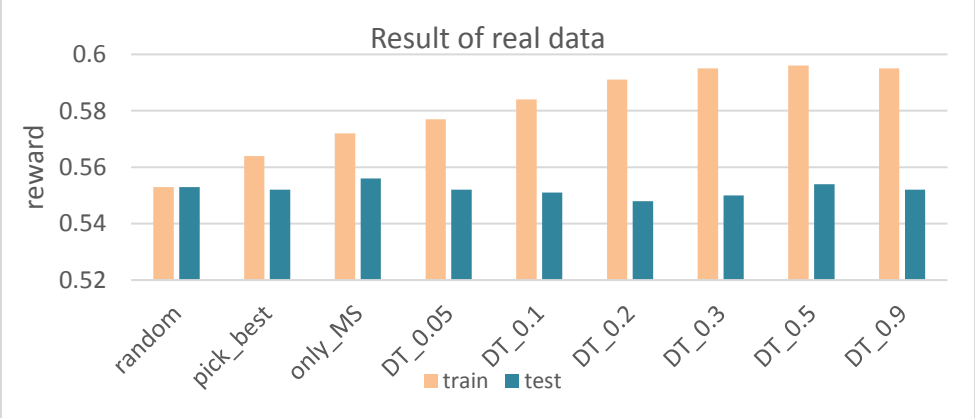


Figure 35. Results of running our decision tree algorithm on 7 features with different significant levels on real ASSISTments experimental data sets, compared with the decision tree constructed with only one feature, prior master speed, and another two methods, random selection and picking the overall best option.

Another issue we want to focus on is the structure of the decision trees. In all constructed trees, the maximum depth is 5 and the minimum depth is 2. These trees have the same structure on the top 2 levels: the feature f1 is used in the root node, and f2 is used in both of nodes in the 2nd level. The mean of all cut points of f1 is 0.491, the mean of cut points of f2 in the left node is 0.665, and 0.685 in the right node. The trees have the structure that is similar with the pre-defined one, except that

they have more levels since we impute the data with some noise – the other four features generated with uniform distribution.

Real Data

1. Results

The second experiment is deployed on the data set collected from 22 real ASSISTments experiments, as described in Section 5.3. In this work, we used 7 features, prior completion rate, prior percent of correctness, prior master speed, imputed gender, learning rate in previous 3 days, percent of correctness in previous 3 days, and prerequisite performance. As illustrated in Section 5.3, we used these 7 features in only 2 experimental data sets, experiment 293151 and 226210, and the first 6 features in other 20 data sets, because too many missing values of prerequisite performance in the other 20 data sets.

In this experiment, we run our decision tree algorithm with different significant levels, 0.05, 0.1, 0.2, 0.3, 0.5, and 0.9. We also use only one feature, prior mastery speed, to build decision trees with significant level 0.9 on this data. The parameter of minimum size is also set to 20 in the decision tree induction. These methods are also compared with random selection and picking the overall best option.

Each method is evaluated with 5-fold cross validation in each experimental data set, and the average of the 22 results is shown in Figure 35. In the training set, the decision tree with larger significant level performs better, but this also could result in more likely to be overfitting. Such as in the testing set, the decision tree with parameter of significant level 0.05 is better than the other decision trees that are constructed on the same set of features. More interesting, the decision tree based on only one feature, prior master speed, is even better than the ones built with more features, but it is not statistical-significantly ($\alpha = 0.9$) better than picking the overall best. Finally, we can conclude that even though some decision trees might be overfitting, if we use appropriate features and parameters, we could get a better result, at least as well as, than just picking the overall best option for all students.

2. Extracted Rules

As aforementioned, it is easy to extract rules from a decision tree that are used to make personalization for students. Here we will show a decision tree built on one of the ASSISTments experimental data sets.

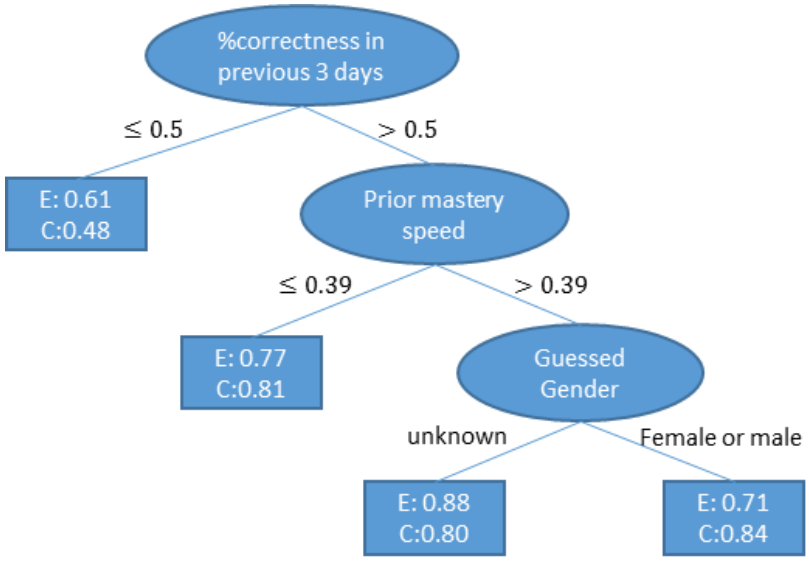


Figure 36. A decision tree constructed on an ASSISTments experimental data.

In this ASSISTments experiment, students were randomly assigned into control group or experiment group. The control group received problems where students had the option to click on a hint button which gave the answer. While the experiment group received problems that did not have the option to click on a hint button, but received help in the form of video buggy messages (Selent & Heffernan, 2015). Students received a short 20-30 second video when they entered a predicted incorrect answer. This video explained what process the student used to arrive at their incorrect answer and how to start on the correct solution path. If a student entered an incorrect answer that was not predicted, a generic message stating that the student's answer was incorrect was shown.

A constructed decision tree on this data set in shown in Figure 36. It is easy to estimate a student's learning outcome, from this tree, in the control group or experiment group by given student's features. To make customization, we can extract following 4 rules from this tree:

1. If a student's percent of correctness in previous 3 days is no larger than 0.5, then assign the student into experiment group;
2. If a student's percent of correctness in previous 3 days is larger than 0.5 and z-score of prior mastery speed is no larger than 0.39, then assign the student into control group;
3. If a student's percent of correctness in previous 3 days is larger than 0.5, z-score of prior mastery speed is larger than 0.39, and unknown guessed gender, then assign the student into experiment group;

4. If a student's percent of correctness in previous 3 days is larger than 0.5, z-score of prior mastery speed is larger than 0.39, and guessed gender is female or male, then assign the student into control group.

5.4.5. Conclusion

We make several contributions in this work. First, with respect to algorithms, we introduce a new discretization algorithm that produces a split value for a continuous feature with the best "cross effect." Furthermore, we implemented a decision tree induction algorithm that can estimate a student's learning outcomes with different choices, and thus make a customization for the student. Second, with respect to evaluating our algorithm, we tested the decision tree algorithm with both simulated and real data, and generate several useful rules. We applied our approach in the context of actual experiments with data generated by real students in a diverse set of middle-school classrooms. While such data are noisy and messy, they provide a much better estimate of the effect of personalization and the impact of algorithm choices than synthetic students generating fake data.

In this section, we have demonstrated the usefulness of our decision tree algorithm in making personalization, and ability of capturing the pre-defined customization structure in a simulated data set. In the next section, I will explain how to combine this decision tree algorithm with bandit algorithms to make decision in the bandit problems.

5.5. Bandits in Decision Tree

To utilize multiple features in bandit problems, we will combine the decision tree algorithm, described in previous section, with bandit algorithms. At each time t_i, we first construct a decision tree on the previous context, $\{x_1, x_2, \dots, x_{i-1}\}$, and observed rewards, $\{r_1, r_2, \dots, r_{i-1}\}$. Then we apply a bandit algorithm for each leaf node in the tree, as illustrated in Figure 37. For current student with context, x_i, we first determine which leaf node it belongs to, and then use the corresponding bandit algorithm to make selection. For example, a student with prior mastery speed 0.3 will be assigned with the Bandit 2 in the Figure 37 to make selection for him/her.

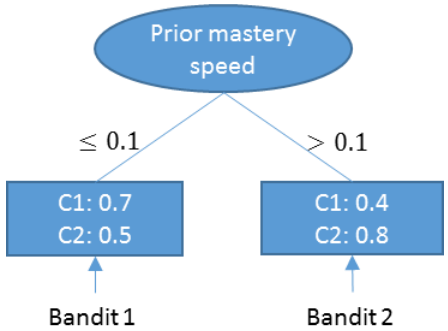


Figure 37. Apply a bandit algorithm for each leaf node of decision tree.

In additional, there are two crucial problems we need resolve in this approach. The first problem is: how to maintain the decision tree structure? The decision tree constructed at current time might produce positive cross effect in the next few time steps, but probably be useless after seeing several new students whose optimal treatments are different from the ones outputted from the decision tree. Therefore, we need to reconstruct the decision tree every few time steps. However, another problem arises here. If we make the reconstruction too frequently, like every round, then it would slow down the process of deciding a selection for a student. In the opposite way, if reconstruct decision tree too infrequently, then the tree might not be reliable. To deal with this dilemma, our approach is to reconstruct the tree frequently at the early time, and increase the frequency when seeing more and more students. The frequency is set to be: reconstruct the tree every student when #students<50; every 10 students when #students<100; every 50 students when #students<500; otherwise, every 100 students.

The second problem is: how to evaluate the contextual bandits? That is, how to estimate the reward for student with a specific condition, especially when the student's real condition is different with the one outputted by the bandit algorithm? In this work, we will use an evaluation process like the one described in Figure 13. For a student s_i at current time with condition c_i assigned by the bandits with decision tree, we first determine which leaf node the student belongs to, and then filter out the set of students, say S_i, in the data falling in the same leaf node. Finally, we pick a random student, rs_i, with the same condition c_i from S_i, and consider the reward of rs_i in the data as the estimated reward of s_i. For example, if a student with prior mastery speed 0.3 is assigned with condition C2 by the bandit with decision tree in Figure 37, then we first locate all students in the data set with prior mastery speed > 0.1, and condition C2, and then pick a random one from those students. The picked student's real reward is assigned to the first student.

5.5.1. Experiments Setup

Bandits to Use

To investigate the effect of context in bandits, we compare the contextual bandits, based on our decision tree algorithm, with UCB1 (Auer, Cesa-Bianchi, Freund, & Schapire, 2002). To be consistent, we will also use UCB1 in the leaf nodes of decision tree. The parameters in the process of decision tree construction are set to be: significant level – 0.3, and minimum size – 20.

Simulation

For each data set, we will simulate running 10 iterations with the two algorithms, UCB1 and DTBandit (bandits with decision tree). At each iteration, the whole data set is defined as an epoch;

while at each epoch, each student in the data set will be inputted into the algorithms one by one, until a certain number of students are tested.

There are two metrics are used in this experiment, average reward and exploration rate. The average reward at each iteration at the time t_i is computed by summing up all previous rewards and then divided by number of students, $\frac{\sum_{j=0}^{i} r_j}{t_i}$. The overall average reward is taking the average of values in the 10 iterations at each time step. To compute the exploration rate, we define if a bandit algorithm picks the condition that is not one with the best average reward at current step, then this bandit makes an explorative selection. Then the exploration rate at each time is the proportion of an explorative selection in the 10 iterations.

5.5.2. Simulated Data

As described in Section 5.4.4, this data set contains 800 instances, 2 different options, option 0 and option 1, and 6 features, 2 of which are used to define distributions that generate the data. The overall better option is option 0, with mean reward 0.53, so the upper bound of bandits without context is 0.53. As shown in Figure 34, the test result of decision tree model with significant level 0.3 is 0.682, since we also set the significant level to be 0.3 in this experiment, so the upper bound of bandits with decision tree is 0.682.

The results of running two algorithms on this data set are shown in Figure 38. We can observe that the performance of UCB1 is approaching its upper bound, and the bandits with decision tree performs better than the UCB1 after about 400 students. However, there is a large gap between the performance of bandits with decision tree and its upper bound. This might be resulted from two reasons: the algorithm still has high exploration rate after seeing many students; the constructed decision tree does not capture the true structure, which results in the optimal treatment outputted from the decision tree is suboptimal in the real data.

Figure 39 shows the smoothed exploration rate of bandits with decision tree on this data. It is computed by taking average of values in 20 neighbor for each data point. From this figure, we can get that the exploration rate after 1000 students is about 0.2 in average, which mean this algorithm has about 20% probability to make explorative selection. This value is not as high as we expected, thus it is not the major reason.

As defined in Table 16, we can regard the structure that defines how to generate the data as a decision tree, in which the root node is feature f1, and its left child and right child at the next level are both f2. Figure 40 shows the proportion of different features, over 10 iterations, used in the

constructed decision tree at each time step. As observed, the decision tree has high probability to use the same feature in the root node after seeing enough students. However, at the next level, it is more likely to use other features than the one, f2, used in the pre-defined structure. This might because the noisy features are effective in some cases. Moreover, as the bandit algorithms would generate options for students that are different with the real options in the original data, and thus results in bias in the reward estimation, which deviates the constructed decision tree from the true structure. Therefore, given that the constructed decision tree cannot capture the true pattern in this experiment, it might be the major reason that causes the performance of bandits with decision tree is much lower than upper bound.

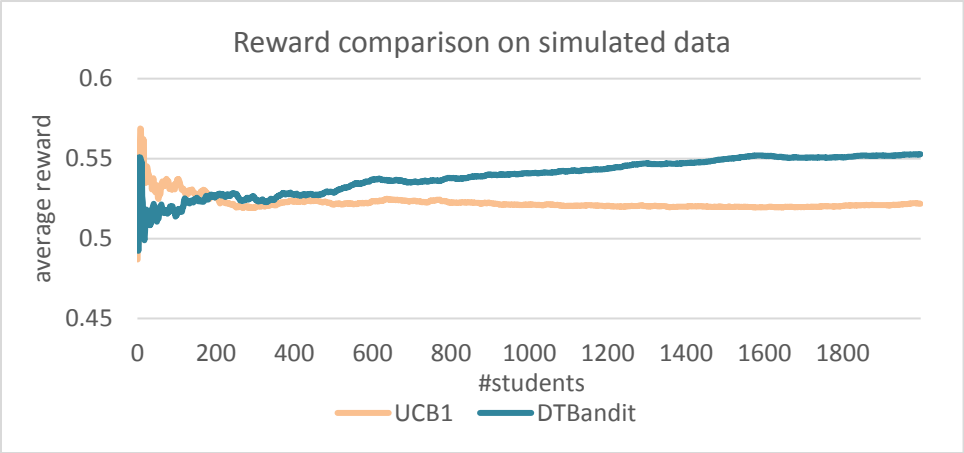


Figure 38. Average reward at each time step of running two algorithms, UCB1 and Bandit with Decision Tree, on the simulated data. Upper bound 1 is the upper bound of bandit without context, and upper bound 2 is the upper bound of bandit with decision tree.

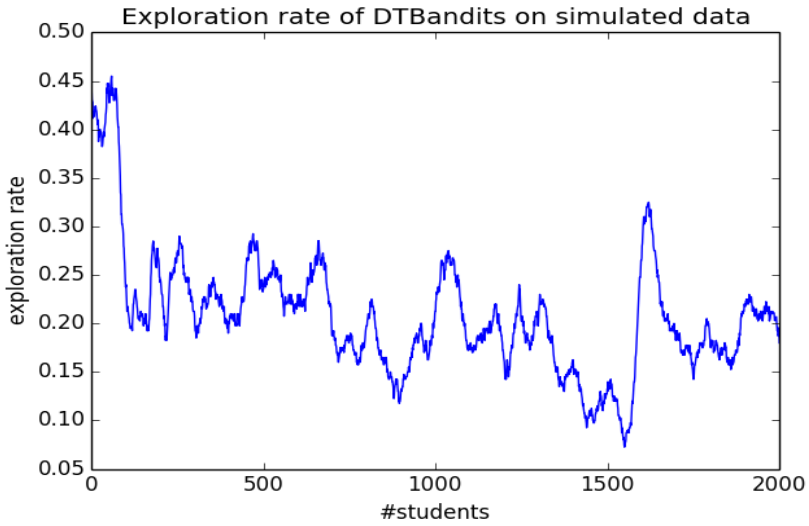
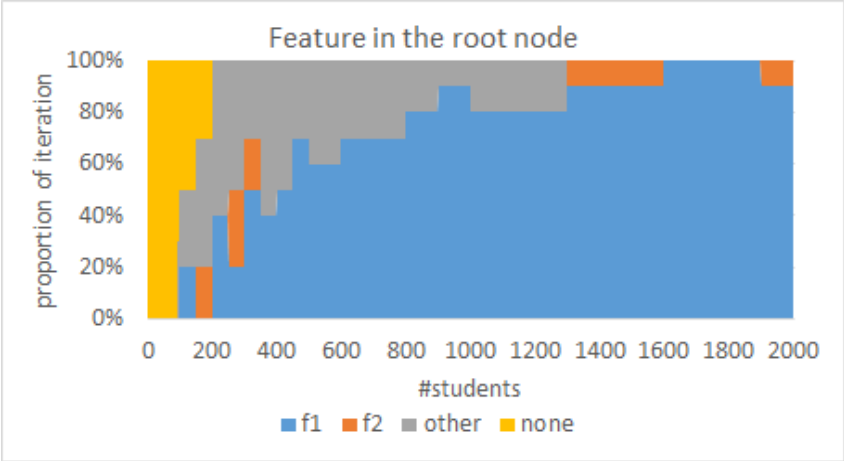
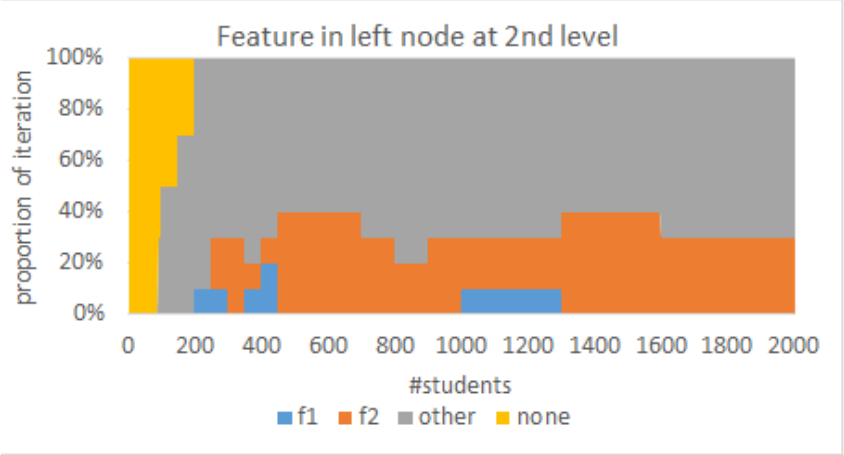


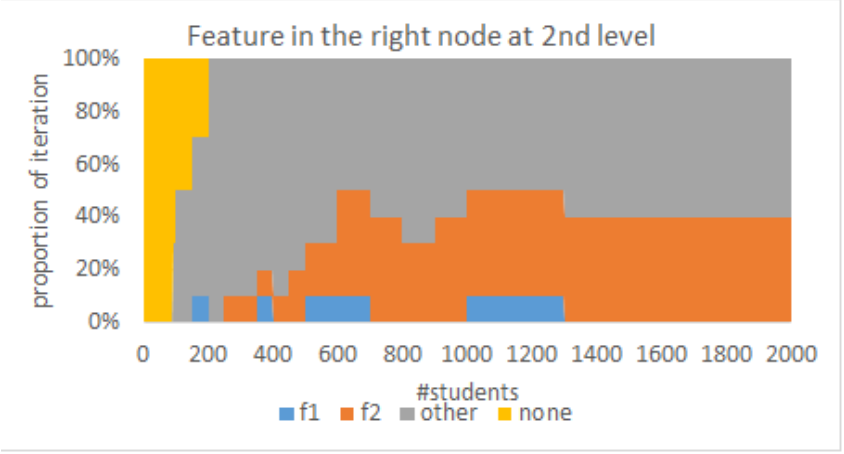
Figure 39. Smoothed exploration rate of bandits with decision tree on simulated data.



(a)



(b)



(c)

Figure 40. Over the 10 iterations in simulated data, the proportion of different features used in the constructed decision tree at each time step. (a) feature usage in the root node of decision trees; (b) feature usage in the left node at the 2nd level of decision trees; (c) feature usage in the right node at the 2nd level of decision trees.

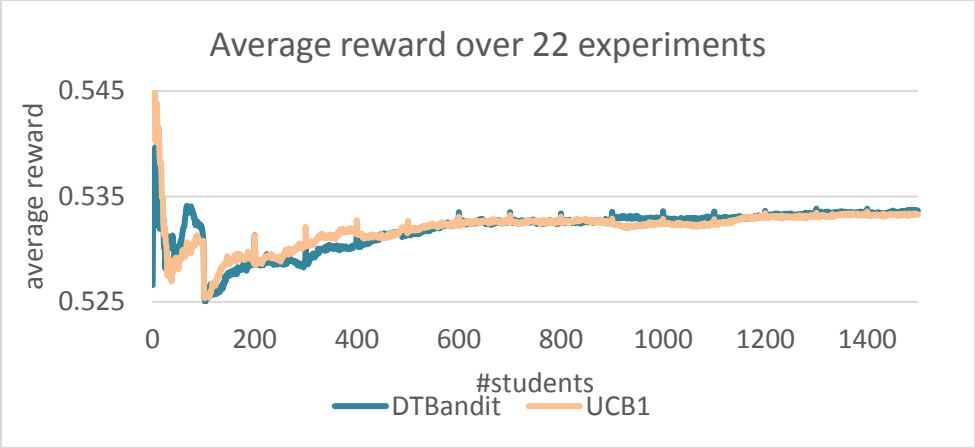


Figure 41. The average reward over 22 experiments at each time step.

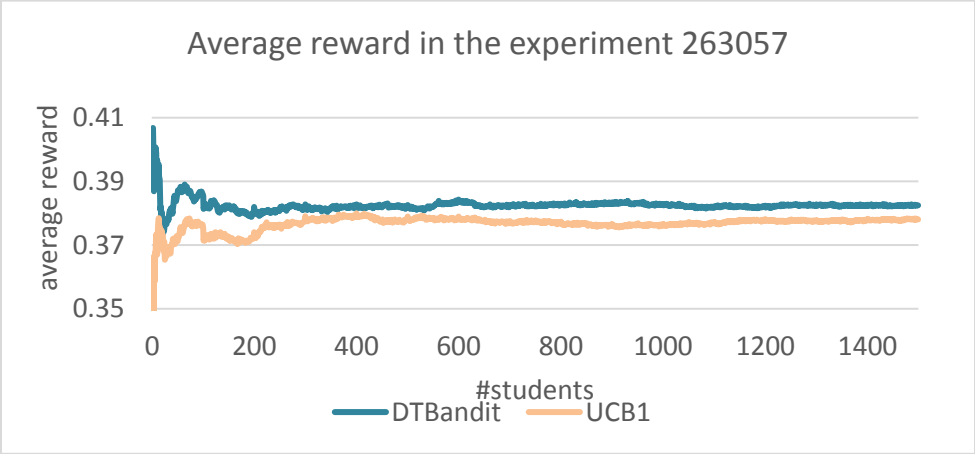


Figure 42. The average reward in the experiment 263057 at each time step.

5.5.1. Real Experimental Data

As described in Section 5.3, this data is collected from 22 real ASSISTments experiments. The experiment with minimum size has 121 students, 1640 of maximum, and there are 10690 students in total. We ran DTBandit with 7 features on the data sets, and then compared it with UCB1 in all experiments. Since the results of decision tree in these data sets, as discussed in Section 5.4.4, did not get significantly better performance than the method of always picking the best, we did not expect the DTBandit would outperform than UCB1 over all experiments.

After running the bandit algorithms on the 22 experiments, we took the average of reward value over all the experiment at each time step for each algorithm. As shown in Figure 41, the performances of DTBandit and UCB1 were very closed to each other, as expected. However, in some individual experiments, DTBandit could obtain better results, like in the experiment 263057, as shown in Figure 42. This might because in some experiments, students' aptitude treatment effect could not be captured by the features, or the decision trees were too overfitted.

CHAPTER 6

Future Works

Future works may focus on different aspects of adaptive learning, like refining student models with skill connections, providing students with diverse types of interventions, and applying k-armed bandit algorithms at problem level.

In this dissertation work, I investigated the effect of prerequisite performance in student models. However, the performance in other skills, not just the pre-required skills, could be reliable factors in student models, especially the skills have strong connections. For example, the skill square root and squaring are related with each other. Therefore, one future work would be to form a mechanism to compute the connection between skills and incorporate such connections into student models. Moreover, we can use such mechanism to evaluate a pre-defined skill structure or the one from other works (Chen, Wuillemin, & Labat; Scheines, Silver, & Goldin, 2014).

To provide students with diverse types of interventions, a possible future work is to build expandable resources with crowdsourcing techniques. For example, implement a function in the intelligent tutoring system that enables teachers or domain experts, or even students, to write illustrative text, add tutoring videos or web pages for learning specific skills. Another possible future work is to enable students to assess or score the interventions. The assessment results are useful to evaluate educational efficacy of interventions. Therefore, we could use text mining and sentiment mining technology to generate important features from the interventions, and thus use the features to pick the appropriate interventions for students to improve their learning progress.

In this work, we applied bandit algorithms at the problem set level – at each time step, pick the optimal conditional problem set for a student, in order to maximize the overall learning outcomes of a group of students. An alternative approach is to apply the bandit algorithms at the problem level, that is, at each time step, pick the proper related problem, or pick the optimal interventions along with the problem for a student, to speed up learning process, or reduce wheel spinning probability, or enhance score in retention test. Furthermore, it is interesting to use the method to explore other tutoring systems, such as Reading Tutor (Mostow & Beck, 2006). The challenge is to obtain a good data set that are large with a lot of interventions. Are we able to find interesting patterns, and what are the potential gains for customization?

Reference

Agrawal, R., Imieliński, T., & Swami, A. (1993). *Mining association rules between sets of items in large databases.* Paper presented at the ACM SIGMOD Record.

Agrawal, R., & Srikant, R. (1995). *Mining sequential patterns.* Paper presented at the Data Engineering, 1995. Proceedings of the Eleventh International Conference on.

Andrejko, A., Barla, M., Bieliková, M., & Tvarozek, M. (2007). User Characteristics Acquisition from Logs with Semantics. *ISIM, 7*, 103-110.

Antunes, C. (2008). *Acquiring background knowledge for intelligent tutoring systems.* Paper presented at the Educational Data Mining 2008.

Arroyo, I., & Woolf, B. P. (2005). *Inferring learning and attitudes from a Bayesian Network of log file data.* Paper presented at the AIED.

Audibert, J.-Y., Munos, R., & Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science, 410*(19), 1876-1902.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning, 47*(2-3), 235-256.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing, 32*(1), 48-77.

Babaioff, M., Sharma, Y., & Slivkins, A. (2009). *Characterizing truthful multi-armed bandit mechanisms.* Paper presented at the Proceedings of the 10th ACM conference on Electronic commerce.

Baker, R. S. (2007). *Modeling and understanding students' off-task behavior in intelligent tutoring systems.* Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Baker, R. S., Corbett, A. T., & Aleven, V. (2008). Improving contextual models of guessing and slipping with a truncated training set. *Human-Computer Interaction Institute*, 17.

Baker, R. S., Corbett, A. T., Roll, I., & Koedinger, K. R. (2008). Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction, 18*(3), 287-314.

Baker, R. S. J. d., Corbet, A. T., & Aleven, V. (2008). *Improving Contextual Models of Guessing and Slipping with a Truncated Training Set.* Paper presented at the In Proceedings of 1st International Conference on Educational Data Mining.

Beck, J. E., & Chang, K.-m. (2007). *Identifiability: A Fundamental Problem of Student Modeling.* Paper presented at the In Proceedings of the 11th International Conference on User Modeling, Greece.

Beck, J. E., Chang, K.-m., Mostow, J., & Corbett, A. (2008). *Does Help Help? Introducing the Bayesian Evaluation and Assessment Methodology*. Paper presented at the Intelligent Tutoring Systems, Montreal, Canada.

Beck, J. E., & Gong, Y. (2013). *Wheel-spinning: Students who fail to master a skill.* Paper presented at the Artificial Intelligence in Education.

Beck, J. E., & Rodrigo, M. M. T. (2014). *Understanding Wheel Spinning in the Context of Affective Factors.* Paper presented at the Proceedings of 12th International Conference, ITS 2014, Honolulu, HI, USA.

Berlyne, D. E. (1960). Conflict, arousal, and curiosity.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*: " O'Reilly Media, Inc.".

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artificial intelligence, 101*(1), 285-297.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher, 13*(6), 4-16.

Botelho, A., Wan, H., & Heffernan, N. (2015). *The prediction of student first response using prerequisite skills.* Paper presented at the Proceedings of the Second (2015) ACM Conference on Learning@ Scale.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*: CRC press.

Brunskill, E. (2011). *Estimating Prerequisite Structure From Noisy Data.* Paper presented at the EDM.

Brusilovsky, P. (1998). *Adaptive educational systems on the world-wide-web: A review of available technologies.* Paper presented at the Proceedings of Workshop" WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems (ITS'98), San Antonio, TX.

Brusilovsky, P. (1998). Methods and techniques of adaptive hypermedia *Adaptive hypertext and hypermedia* (pp. 1-43): Springer.

Burgos, D., Tattersall, C., & Koper, R. (2006). Representing adaptive eLearning strategies in IMS Learning Design.

Cen, H., Koedinger, K., & Junker, B. (2006). *Learning factors analysis–a general method for cognitive model evaluation and improvement.* Paper presented at the Intelligent tutoring systems.

Cesa-Bianchi, N., & Fischer, P. (1998). *Finite-Time Regret Bounds for the Multiarmed Bandit Problem.* Paper presented at the ICML.

Cha, H. J., Kim, Y. S., Park, S. H., Yoon, T. B., Jung, Y. M., & Lee, J.-H. (2006). *Learning styles diagnosis based on user interface behaviors for the customization of learning interfaces in*

an intelligent tutoring system. Paper presented at the International Conference on Intelligent Tutoring Systems.

Chen, Y., Wuillemin, P.-H., & Labat, J.-M. Discovering Prerequisite Structure of Skills through Probabilistic Association Rules Mining.

Clement, B., Oudeyer, P.-Y., Roy, D., & Lopes, M. (2014). *Online optimization of teaching sequences with multi-armed bandits.* Paper presented at the Educational Data Mining 2014.

Clement, B., Roy, D., Oudeyer, P.-Y., & Lopes, M. (2015). Multi-Armed Bandits for Intelligent Tutoring Systems. *Journal of Educational Data Mining, 7*(2), 20-48.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction, 4*(4), 253-278.

Cree, V. E., & Macaulay, C. (2002). *Transfer of learning in professional and vocational education: Handbook for social work trainers*: Routledge.

Csikszentmihalyi, M., & Csikszentmihalyi, I. S. (1992). *Optimal experience: Psychological studies of flow in consciousness*: Cambridge university press.

D'Mello, S., Olney, A., Williams, C., & Hays, P. (2012). Gaze tutor: A gaze-reactive intelligent tutoring system. *International Journal of human-computer studies, 70*(5), 377-398.

Dagger, D., Wade, V., & Conlan, O. (2005). Personalisation for all: Making adaptive course composition easy. *Educational Technology & Society, 8*(3), 9-25.

De Mántaras, R. L. (1991). A distance-based attribute selection measure for decision tree induction. *Machine learning, 6*(1), 81-92.

Desmarais, M. C., Meshkinfam, P., & Gagnon, M. (2006). Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction, 16*(5), 403-434.

Devore, J. L., Farnum, N. R., & Doi, J. A. (2013). *Applied statistics for engineers and scientists*: Nelson Education.

Embretson, S. E., & Reise, S. P. (2013). *Item response theory*: Psychology Press.

Feng, M., & Beck, J. (2009). *Back to the future: a non-automated method of constructing transfer models.* Paper presented at the Educational Data Mining 2009.

Feng, M., Heffernan, N., & Koedinger, K. (2009). Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction, 19*(3), 243-266.

Fossati, D. (2008). *The role of positive feedback in intelligent tutoring systems.* Paper presented at the Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Student Research Workshop.

Freyberger, J., Heffernan, N., & Ruiz, C. (2004). *Using association rules to guide a search for best fitting transfer models of student learning.* Paper presented at the Workshop on analyzing student-tutor interactions logs to improve educational outcomes at ITS conference.

Gong, Y., & Beck, J. (2015). *Towards Detecting Wheel-Spinning: Future Failure in Mastery Learning.* Paper presented at the Learning at Scale 2015.

Gong, Y., Beck, J., Heffernan, N. T., & Forbes-Summers, E. (2010). *The impact of gaming (?) on learning at the fine-grained level.* Paper presented at the Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS2010) Part.

Gottlieb, J., Oudeyer, P.-Y., Lopes, M., & Baranes, A. (2013). Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences, 17*(11), 585-593.

Guo, Q., & Zhang, M. (2009). Implement web learning environment based on data mining. *Knowledge-Based Systems, 22*(6), 439-442.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter, 11*(1), 10-18.

He, Y., Swenson, S., & Lents, N. (2012). Online video tutorials increase learning of difficult concepts in an undergraduate analytical chemistry course. *Journal of Chemical Education, 89*(9), 1128-1132.

Heffernan, N. T., & Heffernan, C. L. (2014). The ASSISTments ecosystem: building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education, 24*(4), 470-497.

Heylen, D., Vissers, M., op den Akker, R., & Nijholt, A. (2004). Affective feedback in a tutoring system for procedural tasks *Affective dialogue systems* (pp. 244-253): Springer.

Hosmer Jr, D. W., & Lemeshow, S. (2004). *Applied logistic regression*: John Wiley & Sons.

Huang, Y., González-Brenes, J., & Brusilovsky, P. (2014). *General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge.* Paper presented at the Educational Data Mining 2014.

Jonsson, A., Johns, J., Mehranian, H., Arroyo, I., Woolf, B., Barto, A., . . . Mahadevan, S. (2005). *Evaluating the feasibility of learning student models from data.* Paper presented at the Educational Data Mining: Papers from the AAAI Workshop.

Kaplan, C., Fenwick, J., & Chen, J. (1993). Adaptive hypertext navigation based on user goals and context. *User Modeling and User-Adapted Interaction, 3*(3), 193-220.

Käser, T., Klingler, S., Schwing, A. G., & Gross, M. (2014). *Beyond knowledge tracing: Modeling skill topologies with bayesian networks.* Paper presented at the Intelligent Tutoring Systems.

Kay, J., Maisonneuve, N., Yacef, K., & Zaïane, O. (2006). *Mining patterns of events in students' teamwork data.* Paper presented at the Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006).

Kelly, K., Heffernan, N., D'Mello, S., Namais, J., & Strain, A. (2013). *Adding teacher-created motivational video to an ITS.* Paper presented at the Proceedings of 26th Florida Artificial Intelligence Research Society Conference.

Kim, J. W., Lee, B. H., Shaw, M. J., Chang, H.-L., & Nelson, M. (2001). Application of decision-tree induction techniques to personalized advertisements on internet storefronts. *International Journal of Electronic Commerce, 5*(3), 45-62.

Koper, R. (2005). An introduction to learning design *Learning design* (pp. 3-20): Springer.

Koutri, M., Avouris, N., & Daskalaki, S. (2005). A survey on web usage mining techniques for web-based adaptive hypermedia systems. *Adaptable and adaptive hypermedia systems*, 125-149.

Kulhavy, R. W., & Stock, W. A. (1989). Feedback in written instruction: The place of response certitude. *Educational Psychology Review, 1*(4), 279-308.

Lan, A. S., & Baraniuk, R. G. (2016). A Contextual Bandits Framework for Personalized Learning Action Selection.

Langford, J., & Zhang, T. (2008). *The epoch-greedy algorithm for multi-armed bandits with side information.* Paper presented at the Advances in neural information processing systems.

Lee, C. D. (2005). ll Signifying in the zone of proximal development. *An introduction to Vygotsky*, 253.

Lee, C. D. (2005). Signifying in the zone of proximal development. *An introduction to Vygotsky*, 253.

Lee, J. I., & Brunskill, E. (2012). The Impact on Individualizing Student Models on Necessary Practice Opportunities. *International Educational Data Mining Society*.

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). *A contextual-bandit approach to personalized news article recommendation.* Paper presented at the Proceedings of the 19th international conference on World wide web.

Li, S. (2013). *Modeling student retention in an environment with delayed testing.* Worcester Polytechnic Institute.

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An enabling technique. *Data mining and knowledge discovery, 6*(4), 393-423.

Liu, K., & Zhao, Q. (2010). Indexability of restless bandit problems and optimality of Whittle index for dynamic multichannel access. *Information Theory, IEEE Transactions on, 56*(11), 5547-5567.

Liu, Y.-E., Mandel, T., Brunskill, E., & Popovic, Z. (2014). *Trading Off Scientific Knowledge and User Learning with Multi-Armed Bandits.* Paper presented at the Educational Data Mining 2014.

Lopes, M., Clement, B., Roy, D., & Oudeyer, P.-Y. (2013). Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*.

Lopes, M., & Oudeyer, P.-Y. (2012). *The strategic student approach for life-long exploration and learning.* Paper presented at the Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on.

Lu, J. (2004). *Personalized e-learning material recommender system.* Paper presented at the International conference on information technology for application.

Lu, T., Pál, D., & Pál, M. (2010). *Contextual Multi-Armed Bandits.* Paper presented at the AISTATS.

Markellou, P., Mousourouli, I., Spiros, S., & Tsakalidis, A. (2005). Using semantic web mining technologies for personalized e-learning experiences. *Proceedings of the web-based education*, 461-826.

Mathematics terms. Retrieved from http://www.doe.virginia.gov/instruction/mathematics/resources/vocab_cards/math_vocab_cards_6-8.pdf

McGuire, P., Logue, M. E., Mason, C., Tu, S., Heffernan, C., Heffernan, N., . . . Li, Y. (2016). *To See or Not To See: Putting Image-Based Feedback in Question.* Paper presented at the International Society for Technology in Education(ISTE 2016).

Mobasher, B., Cooley, R., & Srivastava, J. (1999). *Creating adaptive web sites through usage-based clustering of URLs.* Paper presented at the Knowledge and Data Engineering Exchange, 1999.(KDEX'99) Proceedings. 1999 Workshop on.

Mostow, J., & Beck, J. (2006). When the rubber meets the road: Lessons from the in-school adventures of an automated Reading Tutor that listens. *Scale-Up in Education, 2*, 183-200.

Muldner, K., Wixon, M., Rai, D., Burleson, W., Woolf, B., & Arroyo, I. (2015). *Exploring the Impact of a Learning Dashboard on Student Affect.* Paper presented at the Artificial Intelligence in Education.

Murphy, K. Bayes net toolbox for matlab. Retrieved from https://github.com/bayesnet/bnt/

Narciss, S. (2013). Designing and Evaluating Tutoring Feedback Strategies for digital learning environments on the basis of the Interactive Tutoring Feedback Model. *Digital Education Review*(23), 7-26.

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review, 4*(4), 251-277.

Ostrow, K., & Heffernan, N. (2014). *Testing the multimedia principle in the real world: a comparison of video vs. Text feedback in authentic middle school math assignments.* Paper presented at the Educational Data Mining 2014.

Oudeyer, P.-Y., & Kaplan, F. (2007). What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics, 1*.

Oudeyer, P.-Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on, 11*(2), 265-286.

Pahl, C., & Donnellan, D. (2002). Data mining technology for the evaluation of web-based teaching and learning systems.

Pardos, Z., & Heffernan, N. (2010). *Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm.* Paper presented at the Educational Data Mining 2010.

Pardos, Z. A., & Heffernan, N. T. (2010). Modeling individualization in a bayesian networks implementation of knowledge tracing *User Modeling, Adaptation, and Personalization* (pp. 255-266): Springer.

Pardos, Z. A., & Heffernan, N. T. (2010). *Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm.* Paper presented at the In Proceedings of 3rd Educational Data Mining Conference 2010, Pittsburgh, PA, USA.

Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R. (2008). Learning styles concepts and evidence. *Psychological science in the public interest, 9*(3), 105-119.

Pavlidis, N. G., Tasoulis, D. K., & Hand, D. J. (2008). *Simulation studies of multi-armed bandits with covariates.* Paper presented at the Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on.

Pavlik, P. I., Cen, H., & Koedinger, K. R. (2009). *Performance Factors Analysis - A New Alternative to Knowledge.* Paper presented at the Proceedings of the 14th International Conference on Artificial Intelligence in Education, Brighton, UK.

Philip Jr., I. P., Cen, H., Wu, L., & Koedinger, K. R. (2008). *Using Item-Type Performance Covariance to Improve the Skill Model of an Existing Tutor.* Paper presented at the Proceedings of the 1st International Conference on Educational Data Mining, Montreal, Canada.

Polson, M. C., & Richardson, J. J. (2013). *Foundations of intelligent tutoring systems*: Psychology Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning, 1*(1), 81-106.

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies, 27*(3), 221-234.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*: Elsevier.

Radlinski, F., Kleinberg, R., & Joachims, T. (2008). *Learning diverse rankings with multi-armed bandits*. Paper presented at the Proceedings of the 25th international conference on Machine learning.

Robbins, H. (1985). Some aspects of the sequential design of experiments *Herbert Robbins Selected Papers* (pp. 169-177): Springer.

Robison, J., McQuiggan, S., & Lester, J. (2009). *Evaluating the consequences of affective feedback in intelligent tutoring systems.* Paper presented at the Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on.

Rollinson, J., & Brunskill, E. (2015). From Predictive Models to Instructional Policies.

Romero, C., Ventura, S., & De Bra, P. (2004). Knowledge discovery with genetic programming for providing feedback to courseware authors. *User Modeling and User-Adapted Interaction, 14*(5), 425-464.

Roscoe, R. D., Snow, E. L., & McNamara, D. S. (2013). *Feedback and revising in an intelligent tutoring system for writing strategies.* Paper presented at the Artificial intelligence in education.

Scheines, R., Silver, E., & Goldin, I. (2014). *Discovering prerequisite relationships among knowledge components.* Paper presented at the Educational Data Mining 2014.

Selent, D., & Heffernan, N. (2015). *When More Intelligent Tutoring in the Form of Buggy Messages Does not Help*. Paper presented at the International Conference on Artificial Intelligence in Education.

Self, J. (1988). *Artificial intelligence and human learning*: Chapman and Hall.

Silva, R. C., Direne, A. I., Marczal, D., Guimaraes, P. R., Cabral, Â. S., & Camargo, B. F. (2015). Adapting Collaboratively by Ranking Solution Difficulty: an Appraisal of the Teacher-Learner Dynamics in an Exploratory Environment. *Intelligent Support in Exploratory and Open-ended Learning Environments Learning Analytics for Project Based and Experiential Learning Scenarios*, 41.

Tang, C., Lau, R. W., Li, Q., Yin, H., Li, T., & Kilis, D. (2000). *Personalized courseware construction based on web data mining.* Paper presented at the Web Information Systems Engineering, 2000. Proceedings of the First International Conference on.

Ting, I.-H., Ouyang, Y., & Zhu, M. (2008). eLORM: learning object relationship mining-based repository. *Online Information Review, 32*(2), 254-265.

Vermorel, J., & Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation *Machine Learning: ECML 2005* (pp. 437-448): Springer.

Vuong, A., Nixon, T., & Towle, B. (2011). *A Method for Finding Prerequisites Within a Curriculum.* Paper presented at the EDM.

Wan, H., & Beck, J. B. (2015). Considering the Influence of Prerequisite Performance on Wheel Spinning. *International Educational Data Mining Society*.

Wan, H., & Beck, J. B. (2015). *Considering the influence of prerequisite performance on wheel spinning.* Paper presented at the Educational Data Mining 2015.

Wang, C.-C., Kulkarni, S. R., & Poor, H. V. (2005). Bandit problems with side observations. *Automatic Control, IEEE Transactions on, 50*(3), 338-355.

Wang, W., Weng, J.-F., Su, J.-M., & Tseng, S.-S. (2004). *Learning portfolio analysis and mining in SCORM compliant environment.* Paper presented at the Frontiers in Education, 2004. FIE 2004. 34th Annual.

Wang, X., Wang, Y., Hsu, D., & Wang, Y. (2014). Exploration in interactive personalized music recommendation: A reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 11*(1), 7.

Wang, Y., & Beck, J. (2013). *Class vs. Student in a Bayesian Network Student Model.* Paper presented at the Artificial Intelligence in Education.

Wang, Y., & Heffernan, N. (2013). *Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes.* Paper presented at the Artificial Intelligence in Education.

Weber, G., Kuhl, H.-C., & Weibelzahl, S. (2001). *Developing adaptive internet based courses with the authoring system NetCoach.* Paper presented at the Workshop on Adaptive Hypermedia.

Williams, J. J., Li, N., Kim, J., Whitehill, J., Maldonado, S., Pechenizkiy, M., . . . Heffernan, N. (2014). The MOOClet framework: Improving online education through experimentation and personalization of modules. *Available at SSRN 2523265*.

Xiong, X., Beck, J. E., & Li, S. (2013). *Class distinctions: Leveraging class-level features to predict student retention performance.* Paper presented at the Artificial Intelligence in Education.

Xiong, X., Wang, Y., & Beck, J. B. (2015). *Improving students' long-term retention performance: a study on personalized retention schedules.* Paper presented at the Proceedings of the Fifth International Conference on Learning Analytics And Knowledge.

Yudelson, M. V., Koedinger, K. R., & Gordon, G. J. (2013). *Individualized bayesian knowledge tracing models.* Paper presented at the Artificial Intelligence in Education.