Personalization and Learning Curve Optimization in Intelligent Tutoring Systems

by

Alireza Farasat

Date of Defense: January 05, 2017

A dissertation submitted
to the Faculty of the Graduate School
of the State University of New York at Buffalo
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Industrial and Systems Engineering
University at Buffalo, SUNY, USA

Dissertation Committee

Advisor:

Alexander G. Nikolaev, PhD
Assistant Professor
Department of Industrial & Systems Engineering
University at Buffalo, SUNY

Committee Members:

Varun Chandola, PhD
Assistant Professor
Department of Computer Science & Engineering
University at Buffalo, SUNY

Moises Sudit, PhD
Professor
Department of Industrial & Systems Engineering
University at Buffalo, SUNY

To my wife, ***Sanaz*** *and my daughter,* ***Ava****,*
who have filled me with their unwavering love, patience, and hope, and who have given me a lot of peace and confidence.

Acknowledgements

I am deeply grateful to those that have supported my graduate experience at University at Buffalo. I would like to express the deepest appreciations to my advisor Dr. Alexander Nikolaev, not only for his continuous support and guidance he offered throughout this dissertation but also for all of his insight and particularly for always urging me to think about the potential impacts of my work on educational systems. I was honored to be advised by him who is a perfect example for me in terms of discipline, personality and enthusiasm in advising and teaching students.

I am also indebted to my committee members, Dr. Sudit and Dr. Chandola, for their support and eagerness to understand the problems. They possess the incredible ability to switch instantly their attention to a new area and shed light on whatever technical challenges I explained to them. I owe a great deal to them for the positive energy and confidence that I received.

Many thanks to the UB Department of Industrial and Systems Engineering for providing a productive, progressive, friendly and safe environment which makes the studying and research enjoyable. Special thanks to my friends whose continuous presence, friendship and kindness have been always a great source of comfort and motivation. They did an amazing job in supporting me throughout this process. There are also many other wonderful individuals Ive been lucky and delighted to meet during my time as a graduate student in both Industrial and Systems Engineering and Computer Science and Engineering departments.

Last but not the least, I would like to thank to my amazing mother and parents-in-low, *Khadijeh*,*Lili* and *Hassan*, for their endless love, support, generosity and devotions, and to my beloved siblings and siblings-in-low for their love, encouragements and belief on me. In the memory of my father, *Mohammad Ali*, who would have been happy to see me to accomplish this work.

Contents

List of Figures

List of Tables

Abstract

This dissertation introduces the socio-technical pedagogical paradigm called "crowdlearning", aimed at STEM students in physical and virtual classrooms into collaborative problem-posing and problem-solving activities. This research also addresses the challenges of developing an Intelligent Tutoring Systems (ITS) such as scalability, modeling of student conceptual learning, personalizing optimal teaching policy by incorporating advances in machine learning and optimization techniques to explain how students learn and how an instructor can promote their conceptual understanding of the course contents. A probabilistic framework for formulating and predicting the dynamics of student learning using new tensor factorization models is proposed and aligned with a model predictive control approach to personalize the optimal teaching policy for each student.

The first part of this study develops a web-based platform to engage STEM students and instructors in creative problem posing and solving activities, wherein the students experience deeper learning by gaining new perspectives on the use of subject-matter concepts and by learning with and from each other. The vision of Crowdlearning is that of a self-sustaining problem-posing and problem-solving environment, where the students of a given subject intermittently take on the roles as (A) the creators of subject-focused problems (problem statements/formulations with answer alternatives, hints, correct answers with explanations, etc.) and (B) problem solvers all on the online platform.

The second part of this research introduces two new synthetic data generating models so that enables one to explore students conceptual learning in more depth. The proposed models benefit from the idea of Bayesian Knowledge Tracing and Latent Factor Analysis, the state-of-the-art models in modeling student learning and predicting their performance. The first model consider student learning as a discrete random variable and uses the idea of Factorial Hidden Markov Model to connect different knowledge components and observations overtime. The second model postulates that student learning dynamics is a continuous quantity captured by a first order difference equation. The student learning models developed throughout this dissertation are tested with this synthetic data.

The third part improves current ITS state of the art model in terms of computational performance and proposes a Parallel Coordinate Descent algorithm in shared memory systems that significantly makes the recently proposed SPARse Factor Analysis (SPARFA) scalable. This new implementation enables student modeling analyses at scale, by presenting a parallel algorithm for performing SPARse Factor Analysis (SPARFA). SPARFA is accepted as a state-of-the-art machine learning approach for estimating student knowledge levels and predicting their performance in not-yet-taken educational tasks. However, with the Likelihood Maximization (LM) formulation behind it resulting in a non-convex optimization problem

with many local minima, the scalability has been a challenge for SPARFA. While the original method employs an alternating optimization approach to approximately solve the SPARFA LM problem, this work employs a Parallel Coordinate Descent (PCD) algorithm parallelized in the shared memory setting. The performance of PCD is evaluated on varied problem instances with synthetic data and is found to successfully solve instance with up to 7.5 Million variables, which makes it suitable to make inference from the data of any real-world Massive Open Online Course.

Moreover, this research enhances the current Latent Factor Analysis models and offers new methods for dealing with student performance prediction problem, framed as a Probabilistic Tensor Factorization problem. The proposed framework introduces new constrained tensor factorization models that enable one to discover meaningful patterns from highly sparse data. The key challenge of modeling and predicting students performance lies in the estimation of their conceptual understanding while recognizing both the temporal dynamics of knowledge acquisition and the differences in individual learning abilities. Responding to this challenge, new constrained tensor factorization models in probabilistic formulation are presented. The first model, called Homogeneous Student Model (HSM), is a probabilistic tensor factorization-based model that specifies student knowledge gain as a set of constraints in the model. The second model is based on a new tensor factorization model that parameterizes personalized student learning process and it is called Personalized Student Model (PSM). The proposed models not only Inferring student conceptual learning results in a Likelihood Maximization formulation encompassing non-convex, constrained optimization problems. To tackle these challenging optimization problems, Alternating Direction Method of Multipliers (ADMM) technique is efficiently implemented and the computational results demonstrate that ADMM outperforms Block Coordinate Descent (BCD). The models feature well-interpretable parameters, and are shown to compare favorably against a competing tensor factorization method called Static Student Model (SSM). The computational study reveals that PSM remarkably outperforms SSM and HSM and exhibits very robust performance.

Finally, this study extends a new framework in student modeling that not only takes into account the effect of each question on student learning but it distinguishes also between the gains obtained form correctly and incorrectly answered questions (i.e. correctly answered questions lead to more knowledge gain). The proposed framework that is founded on top of constrained tensor factorization techniques, models the interconnection between different knowledge components as well. The proposed framework called Learning-By-Solving (LBS), is employed as a predictive model to optimize teaching policy by finding the optimal sequence of questions in a personalized way. Finding optimal teaching policy is still an open question in ITSs. To find personalized optimal teaching policies, the corresponding sequential decision making is formulated as a Reinforcement Learning (RL) problem. The resulted RL problem is computationally intractable due to the fact that each question can be used only once; therefore, a Constrained Model Predictive Control (CMPC) approach is employed to attack the sequential decision making problem. The model predictive control offers an adaptive mechanism to estimate students learning dynamics based LBS and optimize the sequence of questions personalized for each student. The computational results illustrate that LBS predicts students performance very well and the CMPC also significantly boost knowledge gain and personalizes the optimal teaching policy.

Chapter 1

Introduction and Motivation

1.1 Introduction

Over the past few decades, the concerns for equity and student privacy protection have led the modern educational systems towards offering students individual-focused learning experiences, by inhibiting or prohibiting public performance comparisons between peers, both in K-12 and higher education settings. Unfortunately, while properly addressing the education justice concerns, the current practices tend to undermine such strong motivational drivers in learning as human competitive spirit and the sense of unity and synergy in achieving common goals. The traditional approach to educational systems is a major drawback to improving the quality of learning outcomes particularly in Massive Open Online Courses (MOOCs). Several attempts have been made over the past few decades to develop new technologies that provide personalized learning pathways for students (Lan et al., 2014).

Intelligent Tutoring Systems (ITSs) are very reliable solutions and have been demonstrated to be highly effective in boosting students learning gains. Increasingly, these new technologies are mainly based on machine learning algorithms and data mining techniques that extract insightful knowledge from the data of a potentially large number of learners interacting with learning contents (Lan et al., 2014b). However, there exist several conceptual and technical issues in developing personalized learning systems that needs to be addressed carefully where some of these challenges are explained and appropriately addressed throughout this dissertation.

This dissertation offers the new pedagogical paradigm Crowdlearning and advances in Intelligent Tutoring System (ITS) technologies to enable collaborative learning at scale. Crowdlearning, to engage students and instructors into a practice of creative problem posing and problem solving. The vision of Crowdlearning is that of a collaborative environment, where students intermittently take on the roles as (A) the creators of subject-focused problems (problem statements/formulations, hints, correct answers with explanations, etc.), and (B) problem solvers which enables learning in parallel with, and potentially beyond, coursework in higher STEM education.

Several technologies are combined to create an automated support system for Crowdlearning: a web-based front-end is to be supported by a back-end that applies ITS to have students perform creator/solver activities in the right sequence, organize collaborative problem cre-

ation, aggregate evaluations, enable communication/teamwork, predict and assess student performance, and control the learning process (so students receive the right material at a rate they can absorb), while offering instructors the ability to monitor/manage this process. Crowdlearning is conceived with an idea that its implementation will be gradually becoming more autonomous over time, by learning about how humans (participating students and instructors) prefer to use it.

The Crowdlearning platform is intended to provide real-time feedback about the learning progress to both teachers and students, while ensuring privacy protection. The statistics of problem creation and solving are to be summarized and shown in aggregate to the problem creators/solvers and to course instructors. Thus, based on what problems are being created and how successfully they are solved, instructors will get feedback about the students' level of understanding of the material; in turn, students will gain a means to find out in what areas they are lagging behind or getting ahead of the peers.

This research employs also advances in Operations Research and Machine Learning to infer latent variables of students conceptual understanding from highly sparse observations obtained from students interactions with ITS platforms. New personalized Tensor-based factorization frameworks that remarkably enable one to estimate dynamics of student learning based on a probabilistic formulation are introduced. To increase scalability of current ITS platforms, Parallel-SPARFA which significantly improves the state of the art student model called SPARFA is presented. Parallel-SPARFA takes advantage of new multi-core programming technologies and considerably improve the performance of the SPARFA core optimization algorithm and makes it more appropriate for learning at scale.

The main deficiency of the existing ITSs is that they mostly offer predictive insights about students performance (Thai-Nghe, 2011; Thai-Nghe et al., 2010; Lan et al., 2014), rather than prescriptive capabilities to improve students education experience. The dearth of models that could allow one to find an optimal teaching policy and personalize for students, accentuates the difficulty in handling such tasks, and at the same time, calls for filling this gap. The existing works in the area of prediction are notable, however, ITSs platforms that engage students into personalized, self-optimized learning experience need to be more specifically focused. In this research, a model predictive control approach is adapted to not only infer the dynamics of student learning but also to personalize learning contents though optimizing the sequence of materials for each student individually.

1.2 Challenges in Educational Systems and ITSs

The recent emergence and widespread popularity of knowledge hubs and learning portals (e.g., Wikipedia, Coursera, Duolingo, Khan Academy, etc.) serves as clear evidence that educational systems are also witnessing a transition from traditional approaches to technology-supported ones. Personalized Learning Systems (Graf et al., 2012) and Intelligent Tutoring Systems (Graesser et al., 2012) are actively researched environments that employ machine learning, cognitive sciences, computational linguistics and optimization to customize learning experiences of students by tracking / predicting their knowledge states. However, these developments also follow the individual-focused learning paradigm. In fact, no applied advances have been made or proposed so far that would exploit the synergy between the learners,

either in virtual (online) classrooms or within the conventional (offline) educational system.

1.2.1 Lack of Effective Feedback

The educational practices currently adopted in K-12 and college settings in the U.S. and many other developed countries lack a means to enable effective and efficient feedback exchange about material understanding between students and teachers during the learning process. Moreover, the most commonly adopted teaching strategies are not flexible enough to meet the diversity of learning capabilities among students. Indeed, it is a hard-to-resolve challenge: how does one maintain equity and justice in the provided educational services, and at the same time, meet each individual students needs by taking into account their unique learning capabilities? A harder challenge: how does one maintain privacy, and at the same time, allow students to exploit their competitive drive, encourage each other to learn harder, help each other, and creatively contribute to making the learning process more clear, more useful and more engaging?

1.2.2 Lack of Students Innovation and Motivation

According to John Dewey, a prominent American philosopher and educational reformer, thinking is provoked by generating questions, not by generating answers. Questions define tasks, express problems and delineate issues. Much work has been done on the kinds of teacher questions that lead to students learning and making connections (Redfield and Rousseau, 1981), but; moreover, recent research demonstrates that providing students with opportunities to ask the questions involves them in thinking through the issues (DILLON, 1988; Van Der Meij, 1994). Helping students to pose and share questions with each other can help them think more deeply about challenging content (Rothstein and Santana, 2011). As understanding is materialized in student questions, instructors can evaluate their understanding and determine what misconceptions exist. The question of interest is: How do we teach students to think, and how can we track and guide their thinking process?

The objective of this research is to assess the ability of the socio-technical pedagogical paradigm Crowdlearning to engage STEM students and instructors in creative problem posing and solving, wherein the students experience deeper learning by gaining new perspectives on the use of subject-matter concepts and by learning with and from each other. The main idea of machine learning and data mining algorithms as the main engine of ITSs is to extract meaningful patterns or build a model using observed data. In the educational systems, there is a few number of observations mainly based on the students' performance on given activities. For example, if one considers students' performance on given tasks as a matrix, the main obstacle is that a fully observed matrix of interest is impossible particularly in MOOCs. While such recovery is not always possible in general, when the matrix is low rank, it is possible to exploit this structure and to perform this kind of recovery in a surprisingly efficient mannerDavenport and Romberg (Davenport and Romberg)

1.3 Main Contributions

This research introduces the socio-technical pedagogical paradigm "crowdlearning", aimed at engaging students in physical and virtual classrooms into collaborative problem-posing and problem-solving activities. It also addresses the challenges of developing ITSs such as scalability, student conceptual understanding, finding optimal teaching policy and incorporating advanced machine learning and optimization models to explain how students perform in the class. A framework for formulating and predicting the dynamic of learning using new Tensor factorization models is proposed and aligned with a model predictive control approach to find the optimal teaching policy for each students.

1.3.1 Crowdlearning: Advanced ITS Platform

The cornerstone idea behind Crowdlearning lies in crowdsourced problem posing. Problem posing is defined as the process of creating a new problem/question or reformulating given problems based on conceptual content (Mishra and Iyer, 2015b; Silver and Cai, 1996). The underlying assumption of this theoretical approach is that students learn the science content and processes in the context of realizing and investigating a problem, and attempt to apply their knowledge to solve the problem (Duschl et al., 2007).

Problem posing is of central importance in many disciplines including but not limited to mathematics, physics, nursing, etc. (Profetto-McGrath et al. 2004, Mestre et al. 2002, Silver et al. 1996). Although not typically adopted in conventional education, the theory of problem posing signifies the motivational and cognitive benefits of posing new problems by students (Beal and Cohen 2012). Indeed, landmark studies in cognitive science and education suggest the need for teaching / learning environments that motivate students to come up with more and better quality questions (Mishra and Iyer, 2015b; Silver and Cai, 1996). Involving students in creative thinking by generating problems and their solutions supports students systems thinking, enhancing their problem-solving strategies to better prepare them for college and career (Rotherham & Willingham, 2009; Shute & Torres, 2012).

A second theoretical basis for Crowdlearning is peer assessment as a learning tool, which has been increasingly emphasized in education. Palmero and Rodrigues (2012) described several advantages of peer assessment activity: evaluating peer works (1) contributes to student learning processes, (2) increases their motivation to learn, (3) improves their perception of their work quality, and (4) increases their responsibility and satisfaction in the learning process. Peer assessment generally includes an activity that students evaluate, and/or is evaluated by their peers. Zundert et al. (2010) reviewed published studies on peer assessment, and found that most studies on this subject reported positive effects on students domain-specific skills – such as science activities, autobiographical writing, and writing a research proposal – from enabling students to revise their own work based on peer feedback. The assessors have a role in reviewing, summarizing, clarifying, giving feedback, diagnosing misconceived knowledge, identifying missing knowledge, and considering deviations from the ideal (Van Lehn et al. 1995), which, in turn, help them to consolidate, reinforce, and deepen understanding of subject matters as well as specific skills (Topping, 1998). Peer assessment not only has been shown to improve learning and skill development, but also to increase students positive attitudes (Cassidy, 2006): students were happy to be assessed by their

peers and felt that peer assessment had improved their work as well as their knowledge.

1.3.2 Learning at Scale with Parallel Sparse Factor Analysis

To improve current ITS state of the art model in terms of computational performance, a Parallel Coordinate Descent algorithm in shared memory systems that significantly improves the scalibility of the recently proposed SPARse Factor Analysis (SPARFA) framework (Lan et al., 2014) is presented. This new implementation enables student modeling analyses at scale, by presenting a parallel algorithm for performing SPARse Factor Analysis (SPARFA). SPARFA is accepted as a state-of-the-art machine learning approach for estimating student knowledge levels and predicting their performance in not-yet-taken educational tasks (Lan et al., 2014).

However, with the Likelihood Maximization (LM) formulation behind it resulting in a non-convex optimization problem with many local minima, the scalability has been a challenge for SPARFA. While the original method employs an alternating optimization approach to approximately solve the SPARFA LM problem, this work employs a Parallel Coordinate Descent (PCD) algorithm parallelized in the shared memory setting. The performance of PCD is evaluated on varied problem instances with synthetic data and is found to successfully solve instance with up to 7.5 Million variables, which makes it suitable to make inference from the data of any real-world Massive Open Online Course.

1.3.3 New Constrained Tensor Factorization Models

This research improves the state-of-the-art in Sparse Factor Analysis, encompassing the methods for solving the student performance prediction problem, framed as a Probabilistic Matrix Factorization problem by introducing new constrained tensor factorization models. The key challenge of modeling and predicting students' performance lies in the estimation of their conceptual learning while recognizing both the temporal dynamics of knowledge acquisition and the differences in individual learning abilities. Responding to this challenge, new constrained tensor factorization models in probabilistic formulation are presented. The models feature well-interpretable parameters, and are shown to compare favorably against two competing tensor factorization methods.

1.3.4 Personalized Optimal Teaching Policy Model

This work also offers a new framework in student modeling that not only explicitly take the impact of each question into account but it distinguishes also between the gains obtained form correctly and incorrectly answered questions. The proposed model which is founded on top of a constrained tensor factorization model, is employed and the predictive model to optimize teaching policy by finding the optimal sequence of questions personalized to each student. To find personalized optimal teaching policy, models based on Reinforcement Learning (RL) and Model Predictive Control (MPC) are developed. The model predictive control offers an adaptive mechanism to estimate students learning dynamics based a constrained tensor factorization model and optimize the sequence of questions individually for each student.

1.4 Structure of Dissertation

The dissertation is organized as follows: Chapter 2 explains the ideas behind crowdlearning and describes the functionality and features of this platform with the focus on problem posing. In addition, the experiences of utilizing the crowdlearning concepts in some undergrad and grad level courses at Industrial and Systems Engineering department at University at Buffalo are shared.

Chapter 3 reviews state of the art models in ITSs that employ machine learning techniques and explores key features of the recently developed models. It also sheds light on main challenges and gaps that the current models are unable to address.

Chapter 4 offers two models to generate synthetic data in order to test the performance of proposed models. The first data generator is based on a Factorial Hidden Markov Model that extends Hidden Markov Model by incorporating multi latent variables. The second synthetic data mechanism uses extensions of Kalman filter and assumes that conceptual learning of contents can be captured by continuous variables.

Chapter 5 extends a parallel version of SPARFA, the state of the art model in student performance prediction and explains the challenges of current model in large-scale MOOCs. It introduces a parallel coordinate descent algorithm to optimize parameters of SPARFA.

New constrained tensor factorization models are proposed in Chapter 6. These models are able to infer student dynamics of learning and estimate student conceptual understanding form sparse observations.

Chapter 7 is devoted to models that are exploited to find optimal teaching policy. The chapter starts with a new tensor-based model that takes the impact of each question on student learning into account. The rest of chapter explains reinforcement learning formulation and model predictive control configuration to deal with this problem. Finally Chapter 8 concludes the research and highlights opportunities and directions for future studies.

Chapter 2

CROWDLEARNING: Towards Collaborative Problem-Posing at Scale

2.1 Introduction

This chapter presents the socio-technical pedagogical paradigm "crowdlearning", aimed at engaging students in physical and virtual classrooms in creative problem-posing and problem-solving. In this approach, learners experience deeper understanding by gaining new perspectives on the use of subject-matter concepts and by learning with and from each other. The vision of crowdlearning is that of a self-sustaining problem-posing and problem-solving environment, where the students of a given subject intermittently take on roles as (A) the creators of subject-focused problems (problem statements/formulations with answer alternatives, hints, correct answers with explanations, etc.); (B) evaluators of problem quality; and (C) problem solvers. The activities that the students perform in roles (A) and (B) are consensus-driven, wherein students create and "vote" in the problems that help them learn, thereby building "banks" of subject matter problems to use for learning and assessment. While crowdlearning can be adopted as an in-class practice, it is expected to be most useful when implemented as an online platform that will direct collaborative activities across classrooms, campuses and colleges, thus enabling an organized growth and refinement of problem banks for individual academic subjects. Such question banks are primed to become ideal companions for classes around the world and MOOCs, enabling deeper learning and automated assessment Koedinger et al. (2015).

This chapter discusses the motivation of crowdlearning as a teaching practice, the theoretical drivers behind it, and challenges to its implementation and adoption. This work then reports on exploratory investigations that assessed STEM students' abilities to competently pose problems and engage in crowdlearning, shedding empirical insights into the associations between learning outcomes and crowdlearning interventions. Finally, it discusses the immediate plans and long-term goals of the ongoing crowdlearning research.

2.2 Motivation and Promise of Crowdlearning

Over the past decades, the concerns for equity and privacy protection have led our educational system towards offering individually focused learning experiences, by inhibiting / prohibiting public performance comparisons between students. Unfortunately, while properly addressing the education justice concerns, the current practices tend to undermine such strong motivational drivers in learning as the human competitive spirit and the sense of unity and synergy in achieving common goals.

The following challenges are particularly characteristic of higher STEM education. First, instructors typically find it hard to assess the level of conceptual understanding of the material by their audience prior to exams, as STEM students are often reluctant to provide live in-class feedback. Second, the same examples and quiz/exam problems tend to be re-used for instruction and assessment each year because formulating problems at higher cognitive levels (e.g., understanding and application) takes much time, while their relative effectiveness is hard to determine. Third, "curves," i.e., distributions of test scores over all students in the same course, are commonly used for assigning grades, yet over the course of a semester, students might not know where they are in the "curve."

The aforementioned challenges played the major roles in the conceptualization of crowdlearning. First, crowdlearning is envisioned as a self-sustaining process of assembling subject-specific problem banks, where products of learning of earlier participants become materials for learning of later participants. In enabling this practice, crowdlearning looks to bring the benefits of "learning from each other" back into today's classrooms, without compromising justice aspects. Online implementations of crowdlearning are intended to provide real-time feedback about the learning progress to both teachers and students, while ensuring privacy protection: the statistics of problem creation and solving are to be summarized and shown in aggregate to the participants. Based on what problems are being created and how successfully they are solved, instructors will get feedback about the students' level of understanding of the material; in turn, students will gain a means to find out in what areas they are lagging behind or getting ahead of their peers.

The idea of crowdlearning stems from the pre-existing explorations of the practice of "problem-posing," in particular the subset of the problem-posing research where learners create "problem" not as open-ended questions but as statements/formulations with answer alternatives (including designated correct one(s)), solutions, hints, etc. In addition to relying on problem-posing to foster deeper learning, crowdlearning exploits the utility of crowdsourcing, where the assignment of the tasks to multiple workers allows even non-experts to collectively produce high quality outputs; indeed, William et al. reported positive outcomes of generating high quality explanations for problem solutions by crowdsourcing this task to the learners Williams et al. (2016). Finally, crowdlearning is primed to rely on and extend the state of the art of intelligent tutoring systems. Given a large problem bank for any academic subject, crowdlearning is intended to employ machine learning and optimization algorithms to personalize the learning experiences of students, both in creating new problems and solving the previously contributed ones.

2.3 Theoretical Bases of Crowdlearning

Postsecondary STEM education often lacks sufficient research-based foundations Braha and Maimon (1997). The historical approach to curricula has been grounded in basic science models, rather than on models for learning. New approaches to STEM teaching demonstrate that students' learning can be stimulated by engaging in the processes of question-posing, peer assessment, and metacognition through game-like activity (e.g., Brindley and Scoffield (1998); English (1998); Li and Tsai (2013); Purchase (2000); Topping and Ehly (2001); Veenman (2012)). These three theoretical/empirical bases grounding this project are taken up below.

The cornerstone idea behind crowdlearning lies in crowdsourced problem posing. Problem posing is defined as the process of creating a new problem/question or reformulating given problems based on conceptual content Mishra and Iyer (2015a); Silver et al. (1996). The underlying assumption of this theoretical approach is that students learn the science content and processes in the context of realizing and investigating a problem, and attempt to apply their knowledge to solve the problem Duschl et al. (2007).

Problem posing is of central importance in many disciplines including but not limited to mathematics, physics, nursing, etc. Profetto-McGrath et al. (2004); Mestre (2002); Silver et al. (1996). Although typically not adopted in conventional education, the theory of problem posing signifies the motivational and cognitive benefits of this activity on students Beal et al. (2012). Involving students in creative thinking by motivating them to generate problems and their solutions enhances their problem-solving strategies thereby better preparing them for college and career Take (2012); Shute and Torres (2012). Moreover, existing studies investigating the relation between the success of students in problem posing and solving show that there exists significant correlation between them Kar et al. (2010). Such activities promote a spirit of curiosity, as well as boosting students' more flexible thinking and fundamental understanding Lavy and Shriki (2010); Lavy and Bershadsky (2003).

However, the research review reveals that the problem-posing concept has been widely utilized as a teaching strategy mainly in mathematics. Meanwhile, it has seen only very limited use in physics, nursing, biochemistry, and computer science and engineering Profetto-McGrath et al. (2004); Mestre (2002); Mishra and Iyer (2015a). It is also worth mentioning that implementing problem posing in traditional class environments ideally requires the use of digital technologies.

A second theoretical basis for crowdlearning is peer assessment as a learning tool, which has been increasingly emphasized in education. Palmero and Rodrigues described several advantages of peer assessment activity: evaluating peer works (1) contributes to student learning processes, (2) increases their motivation to learn, (3) improves their perception of their work quality, and (4) increases their responsibility and satisfaction in the learning process Ruiz Palmero and Sanches Rodriguez (2012). Peer assessment generally includes an activity that students evaluate, and/or is evaluated by their peers. Van Zundert et al. reviewed published studies on peer assessment, and found that most studies on this subject reported positive effects on students' domain-specific skills - such as science activities, autobiographical writing, and writing a research proposal - from enabling students to revise their own work based on peer feedback Van Zundert et al. (2010). The assessors have a role in reviewing, summarizing, clarifying, giving feedback, diagnosing misconceived knowledge,

identifying missing knowledge, and considering deviations from the ideal Van Lehn et al. (1995) which, in turn, help them to consolidate, reinforce, and deepen understanding of subject matters as well as specific skills Topping and Ehly (2001). Peer assessment not only has been shown to improve learning and skill development, but also to increase students' positive attitudes Cassidy (2006): students were happy to be assessed by their peers and felt that peer assessment had improved their work as well as their knowledge.

The third theoretical basis for crowdlearning is an established need to develop students' metacognitive abilities. STEM knowledge theorists claim that students' conceptual understanding is composed of many elements of knowledge, which are spontaneously and unconsciously activated when explaining different phenomena (e.g., diSessa (2002)). Conceptual change is theorized as a process of knowledge integration, based on accumulation, linking, connecting, and structuring of isolated elements of knowledge and concepts Linn et al. (2003). However, the transition to scientific understanding of content does not occur autonomously, since students often do not have metacognitive awareness of their own understanding. Metacognition is this ability to recognize one's current knowledge level and decide to remedy that when it is not adequate Bransford et al. (1999).

Research evidence provides support for the role of metacognition in helping students learn scientific concepts White and Frederiksen (1998). Veenman presents a description of activities congruent with metacognitive teaching and learning approaches: (1) finding activities that promote analyzing the task assignment, activating prior knowledge, goal setting, and planning at the onset of task performance; (2) systematically following a plan or changing, monitoring, and checking that plan during task performance to guide the execution of the task; and (3) evaluating performance against the goal and reflection on the learning process at the end of task performance Veenman (2012).

Taken together, the theoretical and empirical work on metacognition, peer evaluation, and problem posing provides the bases for crowdlearning.

2.4 The First Crowdlearning Platform

In order to assess the potential of crowdlearning to accelerate learning, the authors set out to conduct an exploratory feasibility study aimed to establish the students' ability to competently act in roles (A) and (B) in crowdlearning, i.e., collaboratively initiate the creation of problems and refine them to the point where the instructor would approve their entry into a problem bank.

To this end, an online platform was designed at the Department of Industrial and Systems Engineering at University at Buffalo (SUNY). The platform back-end is supported by MySQL database and PHP, and front-end is written in JavaScript and HTML5 using Laravel API. This freely accessible platform is currently hosted at "http://crowdlearning.eng.buffalo.edu/". The platform is designed to let each participant register as a student, instructor and/or teaching assistant. The instructor view contains such functionalities as setting up a new course, creating a list of topics, directly revising problems created by students, sending the problems back for revision, and approving them, as well as communicating with students and monitoring their performance. Meanwhile, the registered students can propose new problems, revise them, and upon approval, see the aggregate peer performance in those problems. The

module where students collaboratively evaluate and help improve each other's creations is currently in testing.

Each approved problem is automatically added to the problem bank for the corresponding subject; the problems in the bank can then be randomly selected by the platform to be offered to someone to solve, ensuring anonymity. Crowdlearning provides real-time feedback, meaning that immediately after submitting the answer, the correct/alternative solution is viewable along with statistics of other students' performance on the same problem. Figures 2.1 and 2.2 illustrate the problem-posing and problem solving functionalities of the described crowdlearning platform, respectively (see Appendix A for more details of crowlearning functionality).

2.5 Crowdlearning Experiences in STEM

The feasibility studies, assessing the potential for an adoption of crowdlearning, were conducted in the STEM setting, within the curricula of Industrial Engineering, where the use of technological innovations and new instructional practices is rare and much needed.

The 2015 problem-posing module of crowdlearning was tested in two undergraduate Industrial Engineering courses: in "IE 320: Engineering Economy", 12 students were given 59 problems to solve (all created by the instructor); then, in "IE 374: System Modeling and Optimization- OR II", 42 students had 99 problems to solve, of which they created 43. Two graduate courses also used crowdlearning: in "IE 551: Simulation and Stochastic Models", 29 students had 195 problems to solve, of which they created 192; and in "IE 500: Programming for Analytics", 13 students had 37 problems to solve, of which they created 36. Crowdlearning was also used in an online undergraduate course "IE 320: Engineering Economy" offered by Industrial and System Engineering department at University at Buffalo, where 18 students contributed 37 new problems as an extracurricular activity, with the problem bank seeded with sample problems. Five of the student-generated problems were found particularly creative. Seeded by the material given in certain course modules, the students were asked to contribute problems on the respective topic to a problem bank as part of their homework assignments. The completed problems, with the authors anonymized, were made available to all the students to try and solve in preparation for quizzes and exams. Table 2.1 reports students' engagement in six undergraduate/graduate courses using crowdlearning.

The impact of this problem posing and solving on learning has been established qualitatively, with the vast majority of the students voicing support of this practice as a learning aid. The regression analyses with IE 320 and IE 374 data revealed a positive, statistically significant correlation between the student problem solving performance in crowdlearning and their final course grades, albeit with a low R-squared (see Figure 2.3). This can be in part explained by the observation that, while looking to learn, some students did not try hard to solve problems correctly, but instead, proceeded with almost random answers just to see the problems' solutions and explanations. While clearly exposing an issue of "malpractice", this observation gives reasons to believe that students will enjoy and benefit from evaluating peers' problems, wherein they see both the problem statements and solutions.

In 2016, the students of IE 551 "Simulation" students were asked to work in teams (formed

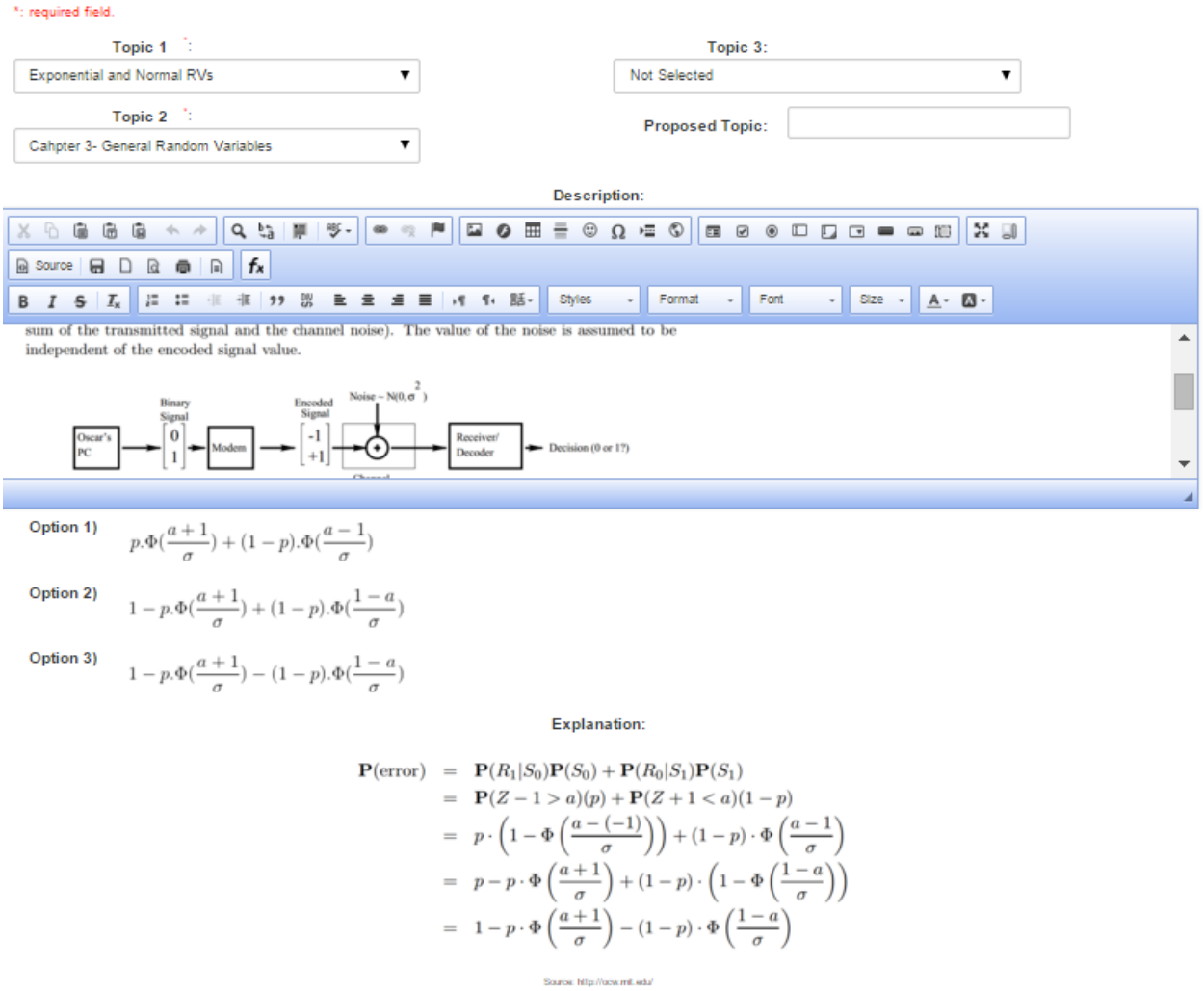

Figure 2.1: Crowdlearning allows students to pose their own problems, which are then given to their peers to solve. Currently, the course instructor or teaching assistant(s) review and evaluate the quality of each student-generated problem and provide an appropriate feedback to the student.

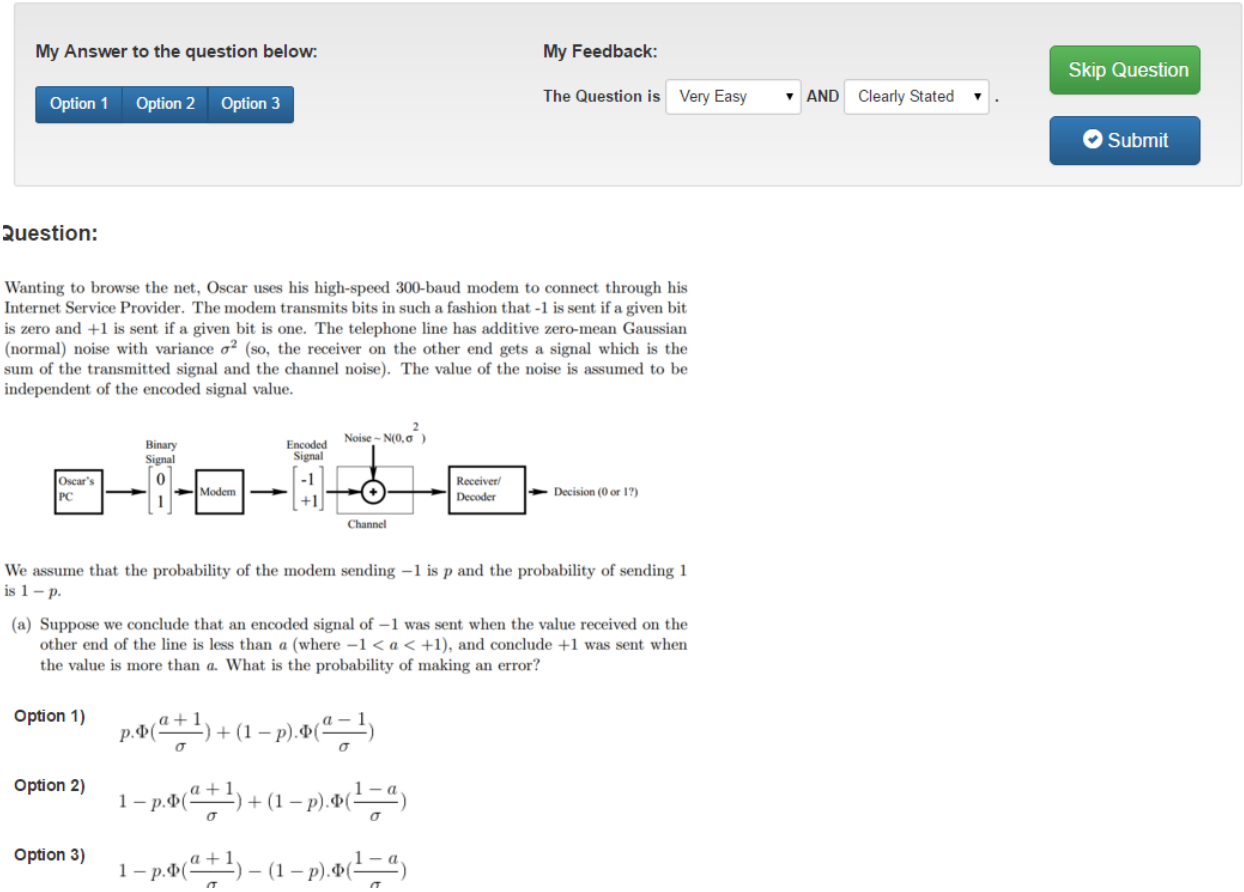

Figure 2.2: Given a problem to solve, a student typically takes 30-50 seconds working on it before providing an answer. In addition, students are asked to provide feedback about the difficulty level and correctness of each problem they work on.

Course	*Students*	*Questions*	*Created by students*
IE 551	29	195	192
IE 500	13	37	36
IE 374	42	99	43
IE 320	12	59	0
IE 575	27	46	0
IE 320*	18	37	5

Table 2.1: A summary of students' participation in crowdlearning for the students in six courses, among which IE 320* was an long-distance (online) course, as one of ISE courses. Columns 2 to 4 report the number of students, the number of questions and the number of questions created by the students, respectively.

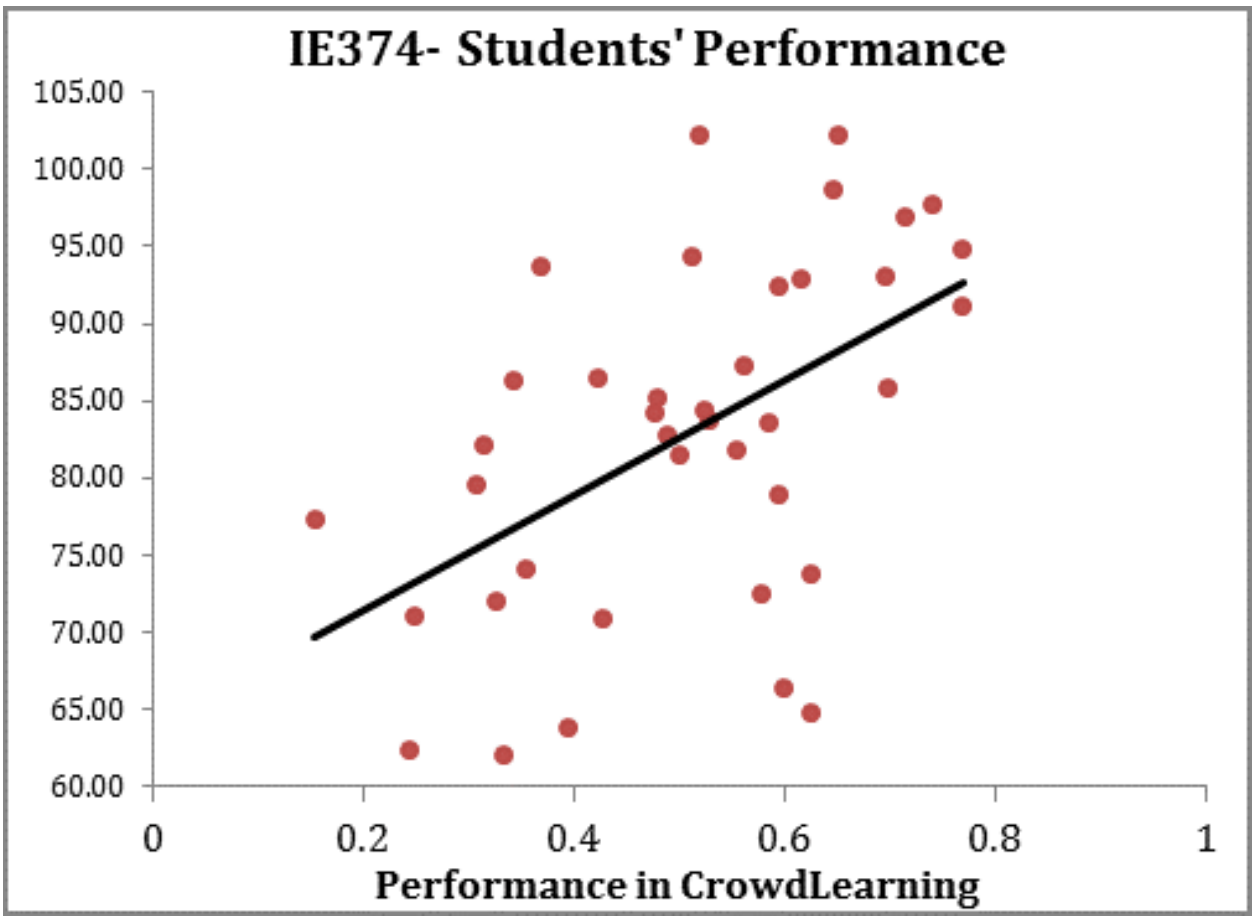

Figure 2.3: A positive correlation is observed between students performance in problem solving activities offered by crowdlearning and their total score in the course. The X axis shows the percentage of crowdlearning questions answered correctly.

Students' Feedback
1) Practice questions on the crowdlearning website are helpful.
2) I loved the practice questions on the crowdlearning website.
3) The homework and crowdlearning assignments really added to my understanding of the course material.
4) I appreciated the instructors' work in trying to get the crowdlearning website up and running.
5) Crowdlearning was helpful in preparing for quizzes/exams because it provides detailed solutions to all questions.

Table 2.2: Excerpts of student feedback on crowdlearning in IE374 and IE320.

at their preference) to pose problems, which, upon the instructor's review and revision, were combined into quizzes for the same student teams to take - each team earned points if they solved a quiz problem right, and also, if their created problem(s) were challenging for the other teams. The instructors contributed 77 problems to seed the problem bank; the student teams then added 81 more problems, of which 41 entered the problem bank after the instructor's review. It was discovered that teamwork leads to both the higher problem quality and higher excitement about the platform use. Many students praised this activity in their anonymous post-course feedback reports, e.g., writing that "...crowdlearning assignments really added to understanding of the course material." (see Table 2.2).

In summary, the described studies established that a majority of college students were capable of posing quality problems on specified subject topics after limited dedicated in-class instruction and with the offline advice provided by the instructors and teaching assistants. This finding is in contrast to the experience of Beal and Cohen who did a similar study with middle school students and reported that the students demanded more flexibility in choosing problem topics and required more guidance Beal et al. (2012), but in line with the experience of Mitros who observed students in an online electronics course to competently create quality problems. The crowdlearning students were also found to be particularly effective in contributing good problem content when working in teams Mitros (2015).

2.6 Future Research and Use at Scale

Connecting the crowdlearning problem bank use with educational materials (textbook excerpts, videos, etc.) can produce comprehensive subject matter knowledge bases and enable the research of individually tailored learning roadmaps, enhancing the learning experiences of those engaged and not engaged in any formal training. In particular, crowdlearning can be used with e-learning tools to enhance the appeal of this form of educational practice. Its dissemination is likely to snowball, as new adopters will be benefitting from all the products (problems bank refinements) contributed by prior adopters; moreover, the problems most helpful to learners will be automatically recognized as such. Crowdlearning opens possibilities for deeper learning in any field where multiple-choice questions can be used for assessment, not limited to the university setting or STEM (e.g., it can be used for SAT preparation). The crowdlearning idea can be expanded to the crowdsourced creation of study "cases" often

used in business, law, etc. Thus, Crowdlearning bridges the gap between crowdsourcing and online learning as envisioned by Williams et al. Williams et al. (2015).

The future crowdlearning platforms will benefit from the advances in intelligent tutoring systems (ITS) to assign tasks to students and guide problem generation in an organized, "smart" manner. The machine learning and data mining algorithms can help us understand how good problems tend to be created. More conventionally, ITS can be useful to explain the dynamics of knowledge acquisition and identifying knowledge gaps Yudelson et al. (2013); van De Sande (2013); predict student performance on not-yet-taken problems Lan et al. (2014,b); identify optimal sequences of roles/activities to assign to each student Rafferty et al. (2011); Clement et al. (2013); and direct crowdsourced evaluation and refinement of new problems.

The next versions of online learning platforms are primed to benefit from the above-mentioned tools and crowdlearning-specific algorithms, e.g., for enabling and improving organized problem assessment. We also envision growing a community of teachers and researchers interested in adopting and developing crowdlearning. Such individuals can serve on voted-in Editorial Boards of instructors who will oversee the formation of publically accessible problem banks.

Chapter 3

State of the Art ITS Models

3.1 Introduction

Explaining, modeling and predicting the learning process as a reflection of human cognition, have been extensively studied in several disciplines such as education, psychology, neuroscience, social science and cognitive science and computer science. From a social science perspective, affect (Linnenbrink and Pintrich, 2004), motivation (Elliot and Dweck, 2013) and identity (Cohen and Garcia, 2008) are three complex macro level interactions affecting student learning (Piech et al., 2015). Moreover, learning is fundamentally considered as human cognition reflection on the micro level (Piech et al., 2015). However, little research has been done to design systems that benefit from the massive amounts of data generated by student transactions with online educational platforms to improve personalized learning experience.

Designing intelligent tutoring systems (ITSs) and computer-assisted instruction dates back to the 1960s. Early efforts of developing such systems concentrated more on approximating the behavior of a human tutor through rule-based systems (Joachims and EDU, 2015). Over the past few decades, several attempts have been made to develop intelligent systems in educational domain using machine learning and data mining techniques (Lan et al., 2014). Today's online education platforms rely more on their ability to gather data at scale compared to the earlier ITSs, which ease the use of machine learning techniques to improve the learning experience (Joachims and EDU, 2015). The main idea of making use of machine learning and data mining algorithms as the main engine of ITSs is to extract meaningful patterns or personalize learning experiences using observed data Khajah et al. (2014).

In this chapter, recently proposed models and algorithms in ITSs are summarized and the most related works in student modeling, especially in describing student learning, predicting student performance and prescribing best teaching policies are reviewed. The first category of models referred to as *Descriptive Models* focuses on explaining the learning process by observing learners' performance. There exist models known as *Predictive Models* with the aim of predicting students' performance over a set of given tasks. The third category of models attempt to optimizing the learning process by maximizing the gain of students' experience on the given tasks. Figure 3.1 depicts the main machine learning-based models

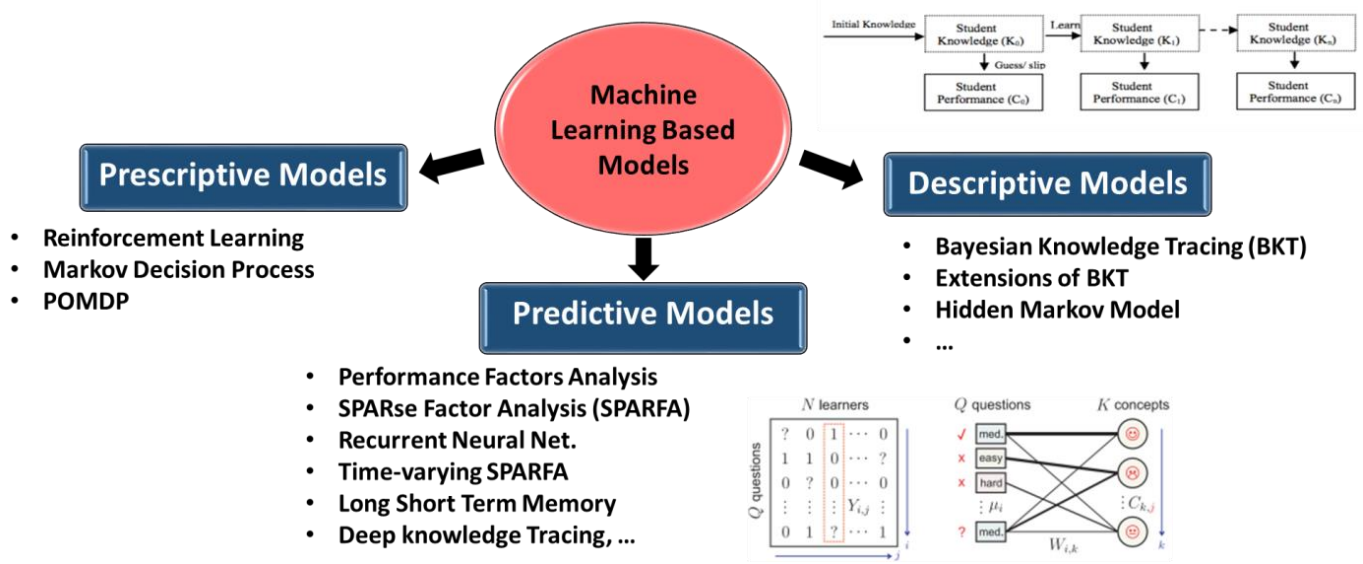

Figure 3.1: Main Machine Learning-based Models in ITSs.

in ITSs.

3.2 Descriptive Models

The ability to use complex series of actions to automatically diagnose student knowledge is becoming more relevant with the increasing use of games and interactive virtual environments in education. Within these environments, students often perform many individual actions to complete a task, resulting in fine-grained data about the choices that students make (Rafferty et al., 2015). Modeling student knowledge as a latent variable is the main idea of Bayesian Knowledge Tracing (BKT) proposed by Corbett and Anderson (Corbett and Anderson, 1994) in 1994. The key assumption in BKT that uses a Hidden Markov Model to model the evolution of student knowledge is that a set of binary variables represents student knowledge (it is assumed that the skill is either mastered by the student or not).

In the original BKT model, it is assumed that students never forget what they have learned which is a very strong, unrealistic assumption. Recent extensions to this model include contextualization of the probability that a student has guessed or slipped on a given question(d Baker et al., 2008), estimating prior knowledge for individual learners through incorporating student-specific parameters into BKT (Yudelson et al., 2013). Another extension of BKT called Feature Aware Student knowledge Tracing, includes the effects of lessons by integrating general features into Knowledge Tracing through an input-output Hidden Markov Model (González-Brenes et al., 2014).

Observations are also binary in BKT illustrating that a student answers a given problem (question) correctly or not (Lan et al., 2013; Yudelson et al., 2013). Although BKT is a powerful technique, it has limitations to model students knowledge state. First, it characterizes learning as a binary variable (either students have mastered or not). In addition, each question is associated with only one concept which is not a realistic assumption. Moreover, BKT

tries to characterize learning process using a single probability of learning. The difficulty level of questions as well as students capabilities are not taken into account in the current BKT models.

3.3 Predictive Models

In the most educational systems, there is a few number of observations mainly based on the students' performance on given activities such as homework, quiz, exam, etc. that can be utilized to infer about students' knowledge and predict their performance. To provide better predictive results, various ensemble methods have been employed to combine BKT and Performance Factors Analysis (PFA) framework (Pavlik Jr et al., 2009). If one considers students' performance on given tasks as a matrix such that each elements of the matrix shows a student's performance on a given question, PFA framework to predict the performance of the student on unseen questions. The main obstacle of this model though is that such a matrix is incomplete and highly sparse particularly in Massive Open Online Courses. While such recovery is not always possible in general, when the matrix is low rank, it is possible to exploit this structure and to perform this kind of recovery in a surprisingly efficient manner (Davenport and Romberg, Davenport and Romberg).

There has been significant research activity aimed at the analysis and development of efficient matrix completion methods (Khajah et al., 2014; Lan et al., 2014b), which seek to impute missing elements of a matrix given possibly noisy or corrupted observations collected at a subset of its locations (Soni et al., 2016). The recently proposed SPARse Factor Analysis (SPARFA) framework (Lan et al., 2014) using probabilistic matrix factorization excels other factor analysis approaches in prediction students' performance (Lan et al., 2014a). Other machine learning based techniques such as RNN model with sigmoid units and a Long Short Term Memory (LSTM) model called Deep Knowledge Tracing have been proposed to the problem of predicting students' performance to based upon their activity history (Piech et al., 2015).

however, these methods are unable to capture dynamics of learning because they postulate that the learners' concept knowledge states remain constant over time.

3.4 Prescriptive Models

There have been substantial in modeling and tracking student learning and predicting students performance as reviewed before. However, there has been much a few attempts on how to find an optimal teaching policy tailored for each student. The optimal teaching policy is obtained by considering the immediate consequences as well as long-term teaching objective using Partially Observed Markov Decision Process (POMDP). Existing POMDP framework for computing an optimal teaching policy assume the learning objective and a set of models describing the learning process are given (Rafferty et al., 2011). Although POMDP provide a reasonable theoretical framework, there exist significant drawbacks to practical implementation such as it requires the learning dynamics and computationally is expensive (Clement et al., 2013; Daubigney et al., 2013; Brunskill and Russell, 2010). Typically, these models have many parameters, and estimating all such parameters for each student is a very

hard problem due to the lack of data, the intractability of the problem, and the lack of identifiability of many parameters (Brunskill and Russell, 2010).

Recent work has explored combining Item Response Theory (IRT) models (Ning et al., 2015) with machine learning based models to provide superior prediction results. Models such as switched extended Kalman filters(Lan et al., 2014b) and Knowledge Tracing Khajah et al. (2014,?). Although these recently proposed approaches are promising, they are restricted in functional form and computational costs which limit the scalablility in MOOCs.

3.5 Deficiencies of Existing Student Models

Building high quality ITS is a complex task that requires not only a certain level of knowledge in multi-disciplinary domains to tack very complicated problems but also relies upon an dealing engineering obstacles in real world implementations. With all advances in the recently proposed models that were reviewed in the previous section, these models suffer from several major shortages that the proposed models in this research directly or indirectly address.

The descriptive models such as Knowledge Tracing and it variations seriously suffer from several issues. In the first place, student conceptual understanding is represented as a binary variable (i.e., either students master the skill or know nothing) in these models which is unrealistic (Piech et al., 2015). Secondly, the concept of the hidden variables in these models and their connections to questions are not well explained, rarely meeting the models expectation of a single knowledge component per question. Although several techniques have been proposed to relieve this issue and refine concept categories and knowledge-question relation, they are able to neither fully capture the mapping between questions and concepts nor represent more realistically student learning. For instance, the current state of the art model, Cognitive Task Analysis (Schraagen et al., 2000) is a laborious and iterative process where domain experts ask students to express their thought processes and share learning experiences during problem solving (Piech et al., 2015).

Although the number of studies on predictive models of student behaviors indicates the important of such topic, the most developed models are unable to infer about student conceptual understanding happening behind the scene. Models such as SPARFA and its variations (Lan et al., 2014b) fail to answer the question how learning happens and what the students' situation is in the learning process and compared to other peers in the course.

In addition, most existing student models mainly offer predictive insights about students performance (Thai-Nghe, 2011; Thai-Nghe et al., 2010; Lan et al., 2014), rather than extending prescriptive abilities to promote students education experience. The lack of models that enabling one to optimize teaching policy and personalize it for students, accentuates the difficulty in handling such tasks, and at the same time, calls for filling this gap. The existing works in the area of predictive models are notable, however, ITSs platforms are needed that engage students into personalized, self-optimized learning experience. In this research, a model predictive control approach is adapted to not only infer the dynamics of student learning but also to personalize learning contents though optimizing the sequence of materials for each student individually.

Finally, many techniques in student modeling employ orthodox tensor factorization mod-

Figure 3.2: 80 percents of pixels are missing in the left figure, and the right figure depicts the low rank approximation reconstruction (Liu et al., 2013).

els which share the same problems with models in recommendation systems. The most common challenges that some of them have been discussed in Recommendation Systems, are closely related to both models and complicity of learning process in human. These challenges are explained and

3.5.1 Cold-start

Cold-start is a well-known problem in recommendation systems that concerns handling new entities of both users and items. Cold-start in ITSs occurs when a new student or new activity is added while there is no historical data or any pattern. Since most ITSs rely upon machine leaning techniques and data-driven models, cold-start is a serious obstacle specially in PFA models. In recommendation systems, content-based models are used to alleviate this issue.

3.5.2 Missing Values

Students' motivation is a critical factor in engaging with ITSs and working on given questions. Only a small subset of questions are answered by students, therefore, remarkable amount of possible student-question interactions remains unobserved. Machine learning-based ITSs usually require tuning parameters that is sensitive to the number of data points in the training set. However, Matrix Factorization based techniques (MF) help to reduce the effect of missing values on the quality of prediction. Figure 3.2 depicts an example of missing value in image processing where 80% of pixels are destroyed and MF models successfully reconstruct the data and estimate missing values.

3.5.3 Partially-observed Feedback

The behavior of students in answering questions is sophisticated and learning is also affected by several uncontrollable variables. In recommendation systems, usually the rate given by each user to some extent represents the true opinion about the item while the performance

of a student may be perturbed by several factors such as forgetting, lucky guess, difficulty of the question, etc. Therefore, models also should be able to consider such noises in analyzing the observation.

3.5.4 Interpretability of Models

In educational systems, it is very important to understand the learning process so that how students master what they have been taught. Hence, one should be able to interpret the parameters obtained by machine learning algorithms. However, many models that offers predictive models are not interpretable such as deep knowledge tracing (introduced by (Piech et al., 2015). The recent model take advantage of neural networks in predicting student performance. However, the parameters obtained from such model do not convey any meaningful message.

3.5.5 Evaluation of Students' Knowledge

One of the biggest challenges in ITSs is to evaluate students' knowledge as hidden variables. The ultimate goal of incorporating artificial intelligence in educational domain is to increase the change that students learn better and faster. In order to evaluate the performance of such models and algorithms, knowing the real state of knowledge that each student is signals not only the quality of an ITS but one can also take advantage of such information to improve the learning process.

3.6 Conclusion

This chapter overviews the state of the art machine learning models developed to describe the learning processes, predict students performance on given tasks and personalize teaching policy commensurate with students' knowledge, learning rate, interests and needs. There are several models introduced to address questions about students learning process, however, one can categorize them based on the features and functionality into three categories. The first category of models known as focuses on explaining the learning process by observing learners' performance. There exist models with the aim of predicting students' performance over a set of given tasks. The third category of models attempt to personalize the learning process by optimizing the gain of students' experience on the given tasks.

Chapter 4

Synthetic Data Generation Models

4.1 Introduction

The ultimate goal of Crowdlearning platform is to promote students' level of knowledge through their performance on given activities. Before developing and implementing any machine learning-based models, a ground-truth is required to enable one to evaluate the quality of the models. The idea of ground-truth data is to provide information about hidden variables related to student's real state of knowledge that none of current platforms are able to provide it, however, such ground-truth data does not exist. Although few data sets have been used in predicting students' performance (Thai-Nghe, 2011), students' conceptual understanding is not captured in these data sets.

Lindsey *et al.* evaluated their proposed model via five student performance datasets varied in the number of students ranged in age from middle school to college, exercises, and skill labels. Each dataset includes student identifiers, trial numbers, exercise identifiers, and binary indicators of response correctness from students over time (Lindsey et al., 2014). To test the performance of SPARFA-Lite, a variation of SPARFA, five datasets collected via OpenStax Tutor[1] have been utilized. The datasets contain undergraduate courses and entrance exam with different number of questions and students in Rice University (Lan et al., 2014a).

Other published data sets extracted from the Cognitive Tutor (used for the Knowledge Discovery and Data Mining Challenge 2010 - KDD Cup 2010)[2],the ASSISTments Platform[3] and Khan Academy[4] have been employed in student modeling mainly in predicting students' performance on given activities. (Feng et al., 2009; Koedinger et al., 2010; Thai-Nghe, 2011).These data sets contain the log files of interactions between the students and the tutoring systems. While the students work on given problems, all their activities, success and progress indicators are stored in databases (Thai-Nghe, 2011).

In most student modeling research, synthetic datasets are extensively used due to lack of required information to evaluate performance of proposed models. Piech *et al.* simulated the process of 2000 students learning five virtual concepts and tested how well they can predict

[1] https://tutor.openstax.org
[2] https://pslcdatashop.web.cmu.edu/KDDCup/
[3] http://teacherwiki.assistment.org/wiki/Assistments 2009-2010 Full Dataset
[4] This is not publicly available

responses on 50 questions given to each student in a controlled setting environment (Piech et al., 2015). Waters *et al.* proposed a a Bayesian approach to the ranked peer grading problem and adopted the model to generate synthetic data. In addition, they developed a procedure for dynamically resolve ambiguity in the data and rapidly resolve a more obvious picture of student performance (Waters et al., 2015).

Nevertheless, the aforementioned datasets are not helpful to study the students fundamental and conceptual understanding of materials taught in the course. Generating synthetic data is not straightforward and needs considerable amount of efforts to model the learning process and parameterize it such that one can translate the underlying dynamics of student learning to a machine learning/optimization problem. In this synthetic data generation, the idea of BKT, SPARFA and extensions of them are integrated. The objective is to model the dynamics of the learning process such that one can track students conceptual understanding through latent variables.

4.2 Model I - Hidden Markov Model Approach

The first model assumes that the conceptual understanding of each knowledge component can be captured by a discrete variable. Let $\mathcal{X}_{kjt}$ denote student j in conceptual understanding of knowledge component k at time t which is a discrete random variable with four possible values such as 0: knowing nothing, 1: weak knowledge, 2: shaky knowledge and 3: solid knowledge.

In BKT model, it is postulated that students either master the skill or know nothing. The original BKT model assumes one knowledge component while variations of BKT relax some of assumptions. In this data generator model, it is assumed that transition happens step by step. In other words, there is no chance that a student master a skill directly form knowing nothing or weak knowledge steps. One can also consider a forgetting factor that leads to transfer from higher states of knowledge to lower states. This is a realistic assumption since many students forget what they have learned a semester after taking the course. Students also have different talents and traits such that some of them are fast-learners and knowledge transition happens very soon while some others require more time to digest the course concepts specially in STEM. Therefore, the model should consider the differences between students.

4.2.1 Probabilistic Model of Knowledge Transition

More importantly, the research problem of interest is the role of each question given to students in promoting students learning and conceptual understanding. Crowdlearning platform is based on the idea of problem posing and providing an collaborative environment that students can share anonymously their questions with other peers. In order to explore the role of such questions, the proposed synthetic data generator distinguishes the impact of each question of knowledge transition. The idea behind this postulation is that questions are not equal in terms of transferring knowledge to students. Some of them play more important role in promoting students conceptual understanding.

Considering all scenarios explained above, the Transition Probability Matrix (TPM) in

(4.1) illustrates the probabilities from one state to other states. Assuming that questions affect students' knowledge differently specially in different time, let $\mathcal{U}_{ijt} = 1(0)$ if question i is given to student j at time t or not.

$$P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}) = \begin{bmatrix} \boldsymbol{1-\rho_{kj}} & \boldsymbol{\rho_{kj}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\rho_F} & \boldsymbol{1-(\rho_F+\rho_{kj})} & \boldsymbol{\rho_{kj}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\rho_F} & \boldsymbol{1-(\rho_F+\rho_{kj})} & \boldsymbol{\rho_{kj}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\rho_F} & \boldsymbol{1-\rho_F} \end{bmatrix}, \tag{4.1}$$

where ρ_F is the probability that a student forgets what she has learned and ρ_{kj} shows the probability that knowledge transition happens for student j in knowledge component k. This probability has two components and calculated as follows:

$$\rho_{kj} = A_{kj} + \mathcal{B}_{kit} \times \mathcal{U}_{ijt}, \tag{4.2}$$

where A_{kj} is the learning rate of knowledge component k by student j and $\mathcal{B}_{ijt}$ indicates how much student j learns by working on question i at time t. One can assume that $\mathcal{X}_{kjt}$ is independent of $\mathcal{X}_{k\prime jt}$, $\forall\ k \neq k\prime$. This assumption significantly simplifies the model, however, this is not a realistic assumption since knowledge components are not independent in many cases. On the other hand, one should consider that fact that increasing the parameters of the models makes estimating parameters from the synthetically generated data extremely difficult. Considering the learning process as a Markov process, the probability that knowledge changes is:

$$P(\mathcal{X}_{:jt}|\mathcal{X}_{:jt-1}, \mathcal{U}_{ijt}) = \prod_{k=1}^{K} P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}) \tag{4.3}$$

The students performance on given questions is a function of knowledge states. In addition, questions difficulty and students' specific parameters are exclusively taken into account. Similar to SPARFA model, students performance is considered as a binary value meaning that students either answer the question correctly (1) or incorrectly (0). Let $\mathcal{Y}_{ijt} = 1(0)$ denote whether student j answered question i at time t correctly (incorrectly) and $P_Y(\mathcal{Y}_{ijt})$ indicate the mass probability function of $\mathcal{Y}_{ijt}$ with Bernoulli distribution with parameter $\phi(\mathcal{T}_{ijt})$ obtained using:

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j, \tag{4.4}$$

where W_{ik} represents the relation between question i and knowledge component k and d_i and θ_j signify the difficulty of question i and student j specific parameter, respectively. $\phi(.) : \mathrm{R} \rightarrow [0, 1]$ can be logit or probit, in this research logit is selected due to simplicity of calculating gradient. Figure 4.1 depicts the corresponding graphical model which is essentially a Factorial Hidden Markov Model (FHMM) (Ghahramani, 2001).

To connect observations with the hidden variables, one can the probability of each observation as follows the distribution defined in (4.5).

$$P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}} \tag{4.5}$$

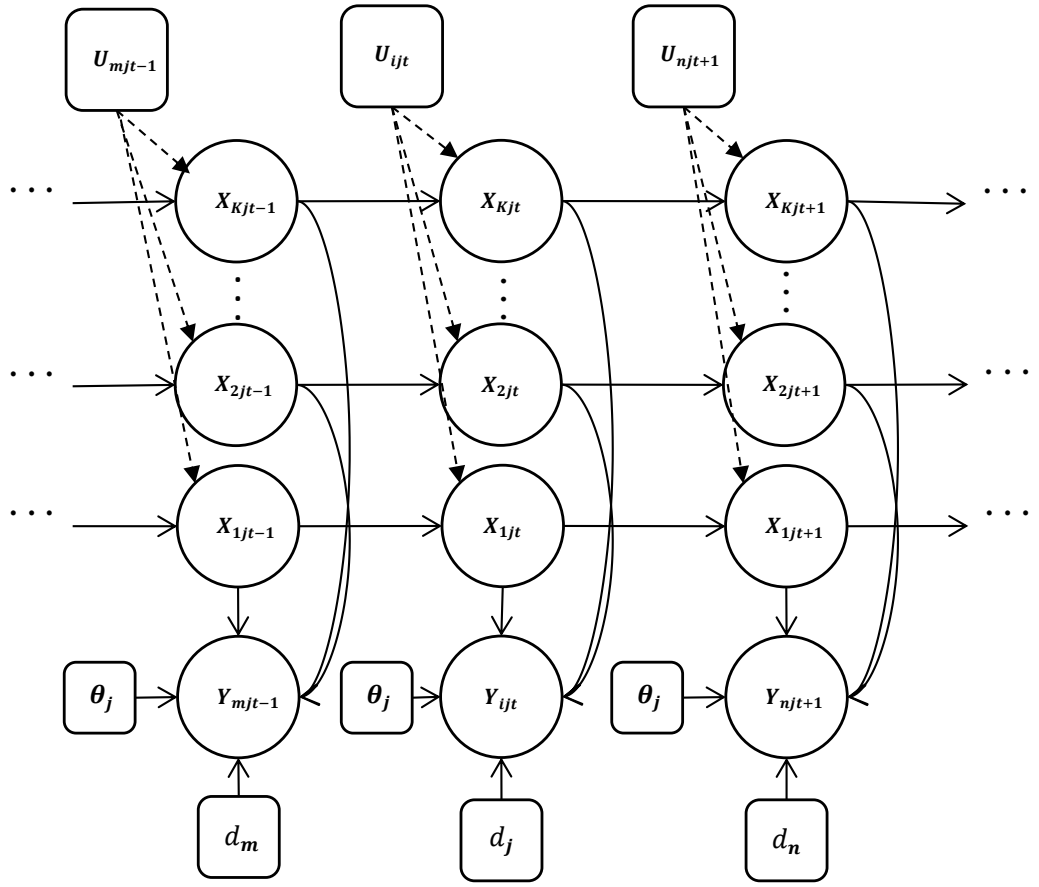

Figure 4.1: Graphical Model representing the underlying dynamics of student j learning and the impact of each question on students' performance.

4.2.2 Experimental Setups

In this section, parameters of the model is tuned and the student knowledge state is explored in more details. First of all, the parameters of Factorial Hidden Markov Model need to be tuned. Three types of questions are considered where have different impacts on student knowledge gains depending on when they are given to students. With equal probability of B_{kit} is one of the following functions given that $W_{ik} > 0$ (if $W_{ik} = 0$, $B_{kit} = 0$):

$$B_{kit} = f_1(t) = \begin{cases} 1 & t < t_1 \\ 1 - 2(\frac{t-t_1}{t_2-t_1})^2 & t_1 \leq t \leq \frac{t_1+t_2}{2} \\ 2(\frac{t-t_1}{t_2-t_1})^2 & \frac{t_1+t_2}{2} \leq t \leq t_2 \\ 0 & t > t_2 \end{cases} \tag{4.6}$$

$$\mathcal{B}_{kit} = f_2(t) = e^{\frac{-(t-c)^2}{2\sigma^2}} \quad c = \frac{t_1 + t_2}{2} \tag{4.7}$$

$$B_{kit} = f_1(t) = \begin{cases} 0 & t < t_1 \\ 2(\frac{t-t_1}{t_2-t_1})^2 & \frac{t_1+t_2}{2} \leq t \leq t_2 \\ 1 - 2(\frac{t-t_1}{t_2-t_1})^2 & t_1 \leq t \leq \frac{t_1+t_2}{2} \\ 1 & t > t_2 \end{cases} \tag{4.8}$$

Algorithm 1 summarizes all steps in generating synthetic data using Factorial Hidden Markov Model.

For example, a machine learning course has three knowledge components including optimization, probability and linear algebra. Parameters for a weak, medium and good student are set. Figure 4.2-4.4 depict the latent variables related to knowledge components status (changes in conceptual learning) overtime for the good, medium and weak student, respectively.

4.3 Model II - Kalman Filter Approach

The second synthetic generation model employs the idea of Kalman filter to produce the data. It's assumed that student conceptual learning is a continuous variable. The core-stone assumption in the proposed synthetic data generator model is the idea that the student conceptual learning is motivated with two primary factors: (i) student interaction with learning resources (e.g.,textbook, lecture video, lab experiment, run a computer simulation, solve a homework, etc.), all of which are likely to result in an increase of their conceptual understanding. (ii) A student may solve a question given by an ITS platform. It is a simplistic assumption that one presumes all questions have the same effect on student learning. Similar to latent factor models, the assumption is that students need to learn distinct K knowledge components to conceptually understand the course materials.

Algorithm 1 Synthetic Data Generator based on Factorial Hidden Markov Model

Input: Tensors $\mathcal{B}, \mathcal{U}$, matrices $\mathbf{W}$, $\mathbf{A}$, vectors $\mathbf{d}$, $\boldsymbol{\theta}$ and parameters N, Q, T, K.
Output: Tensors $\mathcal{Y}$ and $\mathcal{X}$.
SyntheticDataGenerator˙HHM()

1. **for** student $j = 1 : N$
2. **while** $(t \leq T)$ **do**
3. $\mathcal{U}_{ijt}$, $\mathcal{B}_{ijt} \leftarrow$ propose a question
4. **for** knowledge component $k = 1 : K$
5. calculate $P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt})$ using (4.1)
6. given $\mathcal{X}_{kjt-1}$, $\mathcal{X}_{kjt} \leftarrow 0, 1, 2$ or 3 with probability $P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt})$
7. calculate $\mathcal{T}_{ijt} \leftarrow \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j$
8. calculate $P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) \leftarrow \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}}$
9. choose $\mathcal{Y}_{ijt} = 1$ with $P(\mathcal{Y}_{ijt} = 1|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j)$ and 0 otherwise
10. **end**
11. $t \leftarrow t + 1$
12. **end**
13. **end**

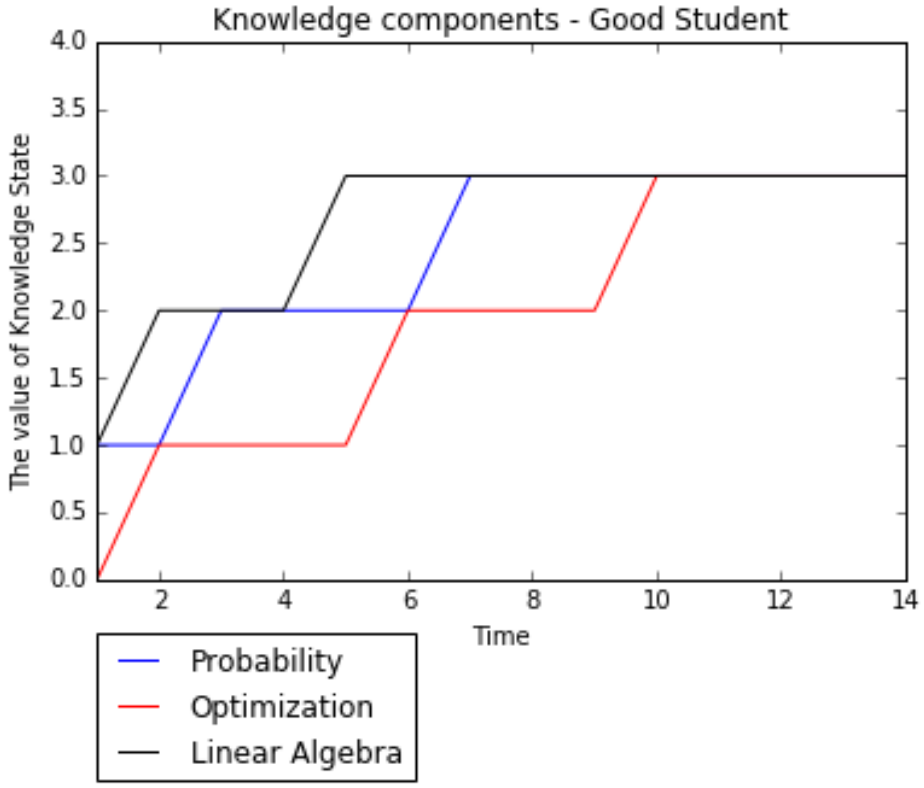

Figure 4.2: Knowledge components of a good student generated by HMM-based model.

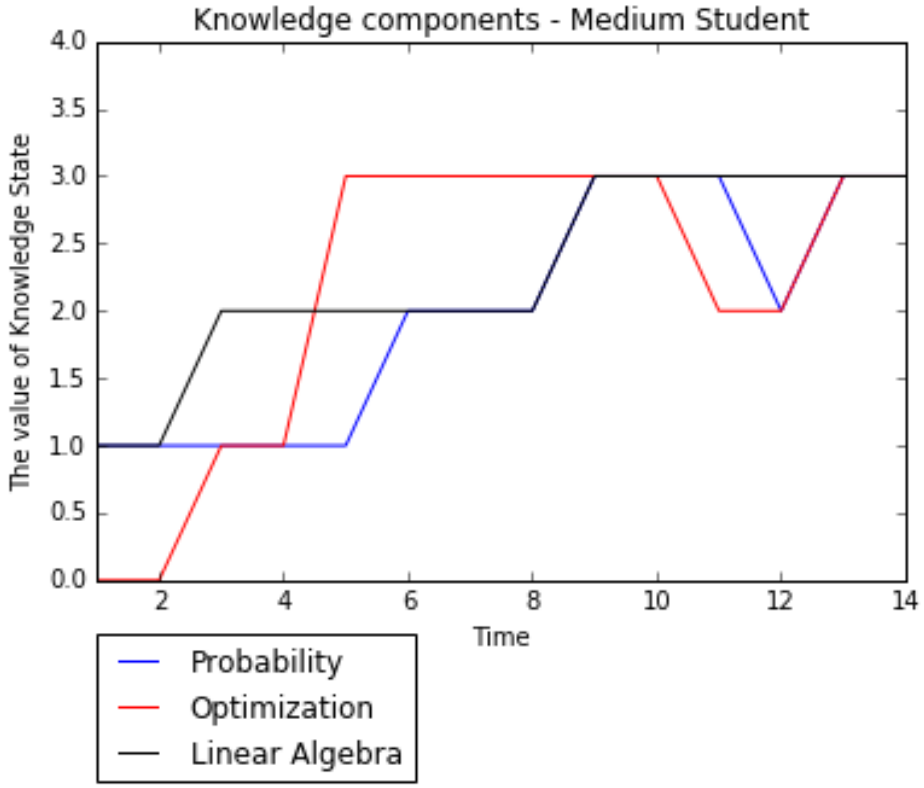

Figure 4.3: Knowledge components of a medium student generated by HMM-based model.

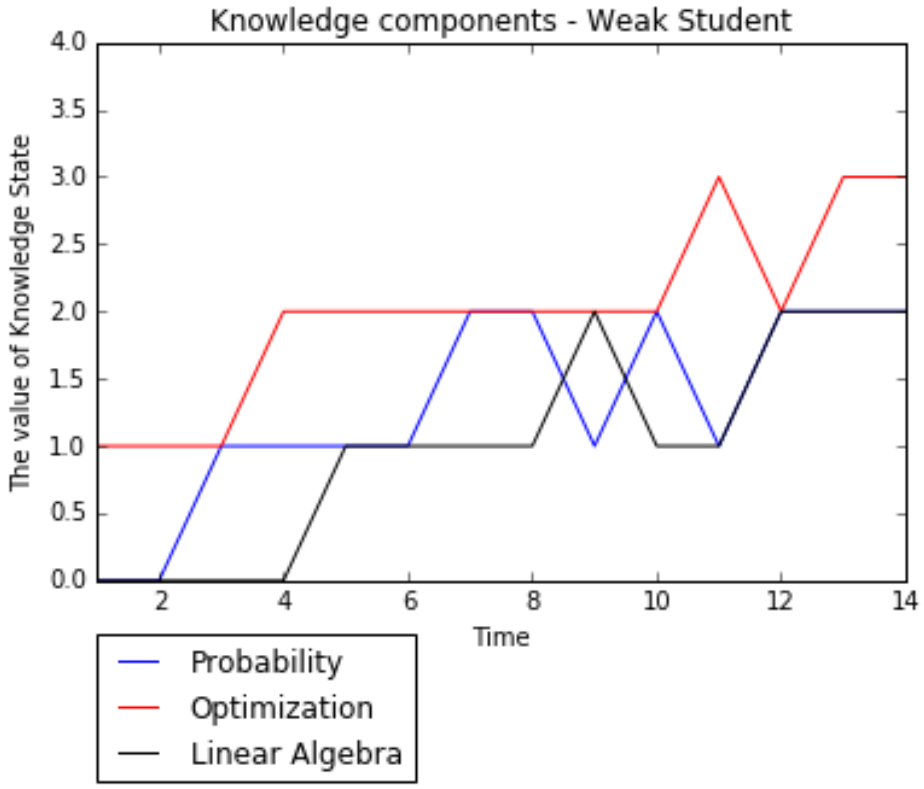

Figure 4.4: Knowledge components of a weak student generated by HMM-based model.

4.3.1 Linear Model of Knowledge Transition

Let $\mathcal{X}_{kjt}$ denote student j's state in knowledge component k at time t and $\mathcal{U}_{ijt} = 1(0)$ denote if the student has solved question i at time t correctly(incorrectly). Assuming Markov property for knowledge gain, student knowledge can be considered as a function of previous knowledge and the question given to the student. Defining such function is consistent with the key assumption about the sources of conceptual understanding, therefore:

$$\mathcal{X}_{kjt} = f(\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}) \tag{4.9}$$

Finding a closed form of (4.9) is a very difficult task specially with sparse data, however, one can replace (4.9) with a linear system that can represent function f.

$$\mathcal{X}_{kjt} = \sum_{l=1}^{K} H_{lk} A_{kj} \mathcal{X}_{ljt-1} + \mathcal{B}^{+}_{kit-1} \mathcal{U}_{ijt-1} + \mathcal{B}^{-}_{kit-1} \Big(1 - \mathcal{U}_{ijt-1}\Big) + \epsilon, \tag{4.10}$$

where $\mathbf{H}$ is a $K \times K$ matrix capturing the interconnection between knowledge components such that $H_{ll} = 1 \quad \forall \quad l = 1, \ldots, K$ and $0 \leq H_{lk} < 1 \quad \forall \quad l \neq k$ while $\mathbf{H} = \mathbf{I}_K$ means that all knowledge components are independent from each other. Student j learning rate in knowledge component k is represented with A_{kj}, where $0 < A_{kj} < 1$. Let $\mathcal{B}^{+}_{kit}$ and $\mathcal{B}^{-}_{kit}$ denote how question i helps one to learn knowledge k at time t if question is answered correctly and incorrectly, respectively. One can define binary variable $\mathcal{U}_{ijt} = 1$ if student j answers question i at time t correctly and $\mathcal{U}_{ijt} = 0$ otherwise.

4.3.2 Probabilistic Knowledge Transition Model

A more realistic approach to knowledge transition is to take uncertainty into account as the source of unexpected factors affecting learning process. The uncertainty of the knowledge gaining is captured by $\epsilon \sim \mathcal{N}(0, \sigma_{kj}^2)$ in (4.10), therefore, the knowledge state of each student is assumed to be a random variable with a Gaussian distribution:

$$\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt} \sim \mathcal{N}\Big(\mathcal{M}_{kjt}, \sigma_{kj}^2\Big), \tag{4.11}$$

where the expected value of student knowledge state is obtained as follows:

$$\mathcal{M}_{kjt} = E[\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}] = \sum_{l=1}^{K} H_{lk} A_{kj} \mathcal{X}_{ljt-1} + \mathcal{B}^{+}_{kit-1} \mathcal{U}_{ijt-1} + \mathcal{B}^{-}_{kit-1} \Big(1 - \mathcal{U}_{ijt-1}\Big). \tag{4.12}$$

In other words, $\mathcal{M}_{kjt}$ is the mean of the distribution. Defining $\mathcal{T}_{ijt}$ as

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ik} \mathcal{M}_{kjt} + \theta_j - d_i \quad \forall \quad i = 1, \ldots, Q \;\; j = 1, \ldots, N, \; t = 1 \ldots T, \tag{4.13}$$

the probability of the observation is:

$$\mathcal{Y}_{ijt} \sim Ber(\Phi(\mathcal{T}_{ijt})),$$

where d_i, and θ_j denote the difficulty of question i and student j specific parameter. *Ber(p)* shows a Bernoulli distribution with success probability p and $\Phi(x)$ is a logit function. The probability mass function of $\mathcal{Y}_{ijt}$ given the parameters is defined as:

$$P_{\mathbf{Y}}(\mathcal{Y}_{ijt}|\mathcal{X}_{:jt}, \theta_j, d_i) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}} [1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}} \tag{4.14}$$

Figure 4.5 depicts the corresponding probabilistic graphical model. In this model, the impact of each question that has been answered correctly or incorrectly is particularly taken into account. For instance, considering a machine learning course with three knowledge components including optimization, probability and linear algebra. Parameters for a weak, medium and good student are set. Figure 4.6-4.8 depict the latent variables related to knowledge components status (changes in conceptual learning) overtime for the good, medium and weak student, respectively.

4.4 Conclusion

This chapter introduces two different models to generate synthetic data. Although there exist few datasets that one can benefit to evaluate the quality of predictions of student performance, they do not offer any clues abut the student conceptual understanding. Therefore, a chapter of this dissertation has been devoted to explain why the data generation process is important and how one can generate data assuring that it is able to capture student learning dynamics.

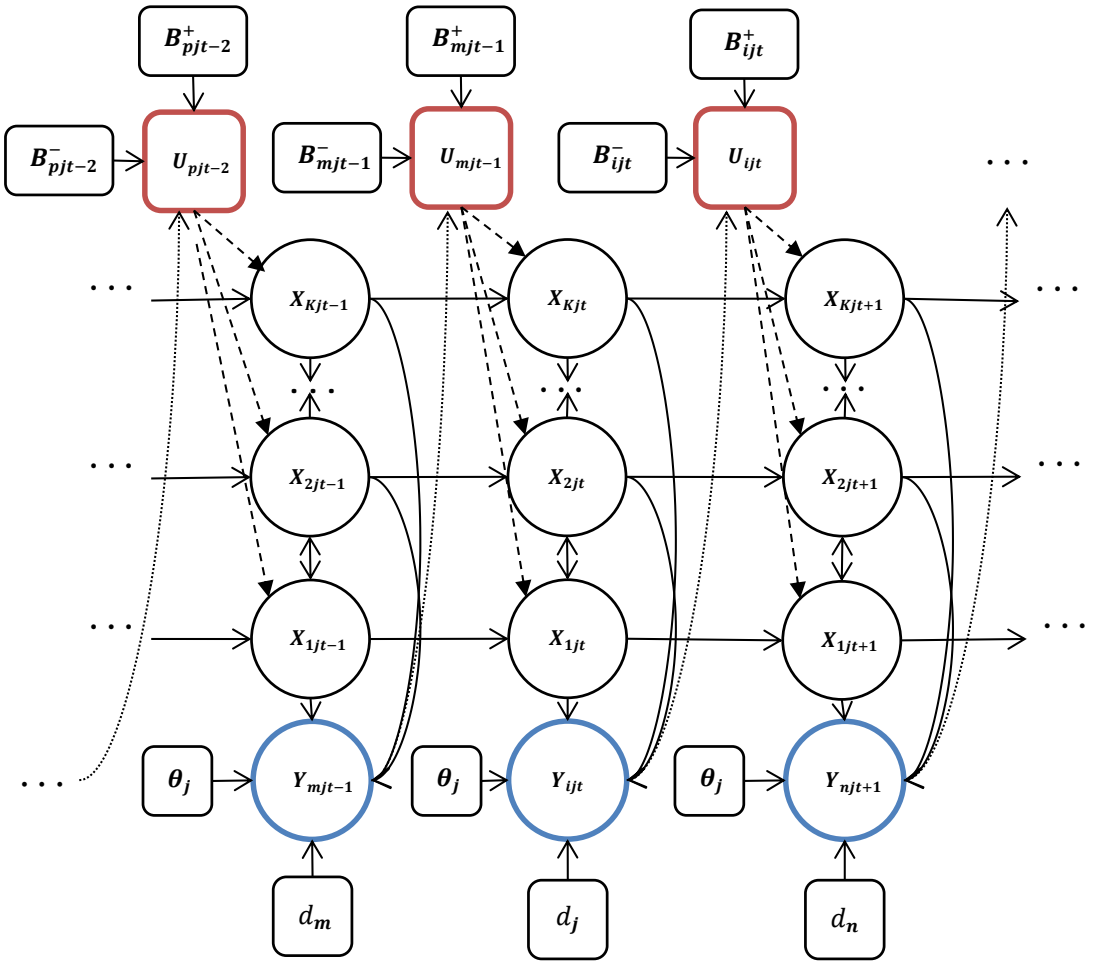

Figure 4.5: Kalman filter graphical model representing the underlying dynamics of student j learning and the impact of each question that the student answered correctly or incorrectly.

Algorithm 2 Synthetic Data Generator based on Extended Kalman Filter

Input: Tensors $\mathcal{B}^+, \mathcal{B}^-, \mathcal{U}$, matrices $\mathbf{W}$, $\mathbf{A}$,$\mathbf{H}$ vectors $\mathbf{d}$, $\boldsymbol{\theta}$ and parameters N, Q, T, K.
Output: Tensors $\mathcal{Y}$ and $\mathcal{X}$.
SyntheticDataGenerator˙HHM()

1. **for** student $j = 1 : N$
2. initialize $\mathcal{X}_{:j0}$
3. **while** $(t \leq T - 1)$ **do**
4. $\mathcal{U}_{ijt}$, $\leftarrow$ propose a question
5. **for** knowledge component $k = 1 : K$
6. calculate $\mathcal{X}_{kjt+1}|\mathcal{X}_{kjt}, \mathcal{U}_{ijt} \sim \mathcal{N}\Big(\mathcal{M}_{kjt+1}, \sigma^2_{kj}\Big)$, using (4.12)
7. calculate $\mathcal{T}_{ijt} \leftarrow \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j$
8. calculate $P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) \leftarrow \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}}$
9. choose $\mathcal{Y}_{ijt} = 1$ with $P(\mathcal{Y}_{ijt} = 1|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j)$ and 0 otherwise
10. **end**
11. $t \leftarrow t + 1$
12. **end**
13. **end**

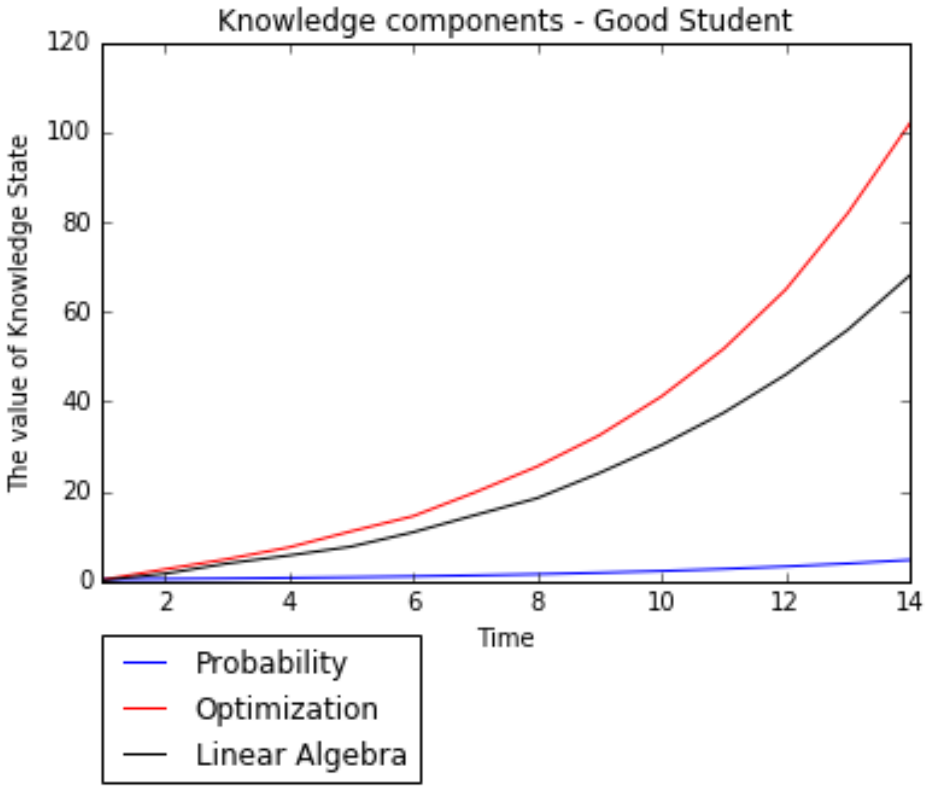

Figure 4.6: Knowledge components of a good student generated by Kalman Filter-based model.

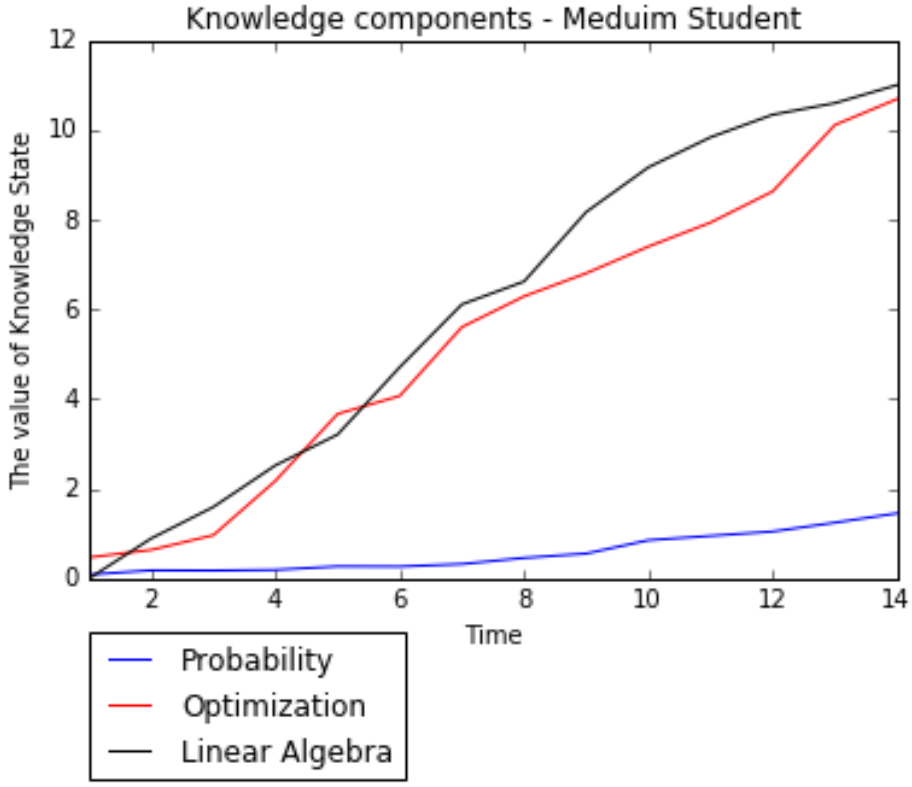

Figure 4.7: Knowledge components of a medium student generated by Kalman Filter-based model.

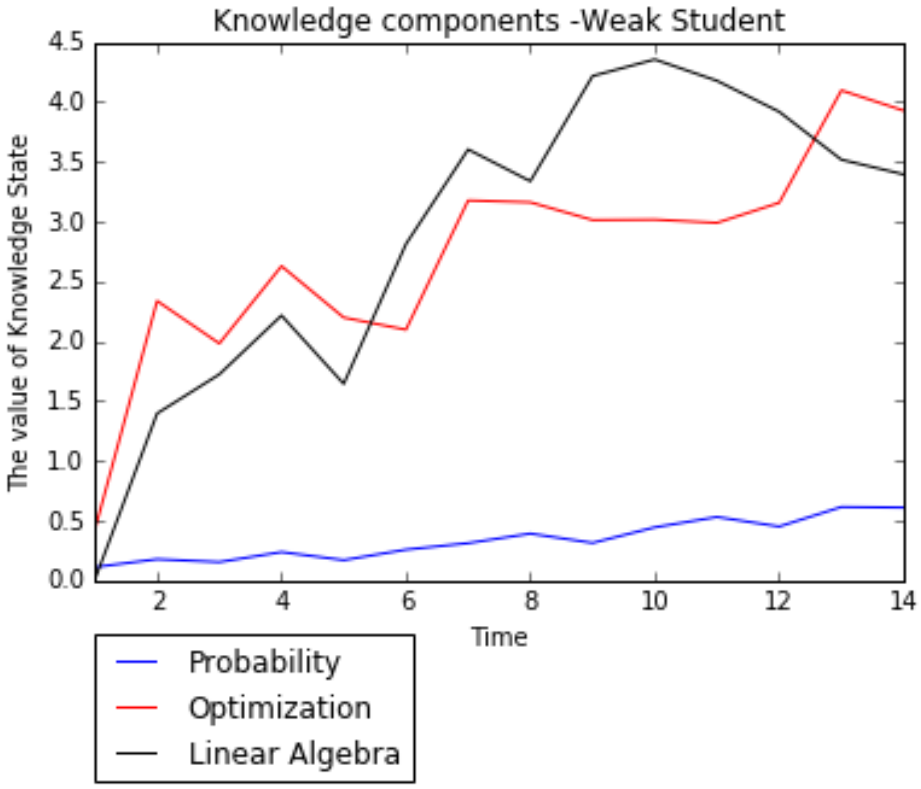

Figure 4.8: Knowledge components of a weak student generated by Kalman Filter-based model.

The first synthetic data generator model is based on the idea of Factorial Hidden Markov models that consider more than one latent variables. This model mimic the learning process and gradual changes in student knowledge state. The basic idea of this model is BKT and SPAFA which are among the best known frameworks in student modeling domain. The second synthetic data generator model concentrates on how students learn by solving given questions and distills the impact of questions correctly or incorrectly answered in the past.

Chapter 5

Parallel Sparse Factor Analysis

5.1 Introduction

Educational systems have witnessed a substantial transition from traditional educational methods mainly using text books, lectures, etc. to newly developed systems which are artificial intelligent-based systems and personally tailored to the learners. The resulting popularity of online educational platforms with large numbers of students and tasks/problems to solve motivates the development of highly efficient algorithms to personalize the learning experiences (Lan et al., 2014). Personalized Learning Systems (PLSs) and Intelligent Tutoring Systems (ITSs) are two areas of educational data mining that take on this challenge. Taking into account the individual progress of learners, PLSs customize the learning experience to the learners' abilities/needs (Graf et al., 2012). In computerized learning environments, ITSs model and infer student's knowledge learning states (Graesser et al., 2012; Rafferty et al., 2011; Sahebi et al., 2014). For a while, Latent Factor Modeling and Bayesian Knowledge Tracing have been the primary student modeling ITS tools (Khajah et al., 2014). These approaches encompass computational models, conceived in different scientific disciplines including cognitive and learning sciences, education, computational linguistics, artificial intelligence, operations research, and other fields Graesser et al. (2012); Rafferty et al. (2011); Li et al. (2011); Lan et al. (2014).

More recently, Lan *et al.* developed a new machine learning-based model that combines *learning analytics*, which approximates students knowledge of subject matter concepts underlying a domain, and *content analytics*, which estimates the relationships among a collection of questions and those concepts Lan et al. (2014). This model can be used to calculate the probability that a learner provides a correct response to a question in terms of three factors: their understanding of a set of underlying concepts, the concepts involved in each question, and each questions intrinsic difficulty (Lan et al., 2014). Lan *et al.* proposed a bi-convex Likelihood Maximization approach for SPARse Factor Analysis (SPARFA), and exploited an alternating optimization approach to approximately solve the LM problem.

A SPARFA solver has to iterate between two sub-problems, until each sub-problem is optimally solved (due to the bi-convexity of the likelihood function). However, in order to run SPARFA analyses with large numbers of questions and students, i.e., to make it useful for MOOCs of realistic sizes, its scalability needs to be improved. To this end, this paper

presents a parallel coordinate descent (PCD) algorithm that uses shared memory. The idea behind PCD is to parallelize the optimization task over computing resources (i.e., cores or processors) so that the cores work on different parts of the optimization problem (in terms of blocks of variables) simultaneously. At each iteration of PCD, a number of variables (i.e. from each block of variables) depending on the number of threads are selected randomly. Each thread individually works on a smaller set of variables and update them based on the gradient information.

This chapter contributes to the educational data mining domain with the resulting algorithm that can handle a very large number of variables – up to 7.5 Million variables – in a given SPARFA instance. The computational experiments with synthetic data exhibit that PCD produces up to 7.4x speedup, with the highest speedups achieved with the largest instances.

The rest of this chapter is organized as follows. It first reviews the original SPARFA model and provides an overview of parallel optimization algorithms proposed for MF applications. It also gives a detailed description of the new PCD algorithm.The fourth section reports experimental results on synthetic data of varied sizes. The last section concludes the chapter and discusses future research directions.

5.2 Challenges of Sparse Factor Analysis at Scale

Matrix Factorization (MF) is a key component of machine learning-based systems that work, e.g., to recommend items to users, estimate missing data values, model gene expressions. More recently, MF has been found useful for predicting students' performance on given activities, e.g., problem solving. Indeed, with the advent of online learning platforms and their massive accessibility, much data are becoming available about students' knowledge states to assist educators in monitoring and accelerating learning. The developments in this paper address the scalability of MF performed for large educational systems, where thousands of students in physical or virtual classes work on the same set of problems referred to as problem bank. This section overviews and outlines the computational challenges of SPARFA as a state-of-the-art machine learning framework for MF in student modeling. SPARFA utilizes an alternating optimization approach to tune the parameters of its model, however, this approach is computationally expensive, especially in large-scale problems. This section argues in favor of an alternative algorithm to use computational resources more effectively via efficient parallelization.

Student modeling is the cornerstone ITS task that reveals how the knowledge acquisition process can be decomposed as progress in multiple dimensions, typically referred to as knowledge components or skills Li et al. (2010); Khajah et al. (2014). In order to strategically attack the problem of assigning educational tasks to students in an personalized manner, one needs to both infer each student's skills and determine which sets of skills are required to handle the tasks not yet undertaken by the student. Factor analysis is a well accepted approach to student modeling based on the student performance in quizzes and homeworks (Lan et al., 2014a);Lan et al. (2014). SPARFA is a machine learning-based model that connects learning analytics and content analytics to students' performance. It infers the probability that a student correctly answers a question or solves a problem based on her

understanding of a set of underlying course concepts, on the concepts involved in each question, and each question's intrinsic difficulty (Lan et al., 2014). Given an incomplete matrix of students' performances, Maximum Likelihood Estimation (MLE) is performed in SPARFA to make the inference about students and questions, simultaneously.

Let $\mathbf{Y}$ denote a binary-valued data set of performance of N students on Q questions; hence, $\mathbf{Y}$ is a matrix of size $Q \times N$ with entry $Y_{ij} = 1(0)$ if student j has answered question i correctly. Matrix $\mathbf{Y}$ is typically very sparse. One way to estimate the missing values in $\mathbf{Y}$ is to factorize $\mathbf{Y}$ into matrices $\mathbf{W}, \mathbf{C}$ and $\mathbf{M}$ such that the function $\mathbf{WC+M}$ returns the estimates for the missing values in $\mathbf{Y}$. It is assumed that the collection of questions is related to a small number of abstract concepts represented by $\mathbf{W}$, where the weight W_{ik} $(\forall i = 1, \ldots, Q \; and \; k = 1, \ldots, K)$ quantifies the degree to which question i involves concept k, with K being a total assumed number of such latent abstract concepts. Let C_{kj} $(\forall k = 1, \ldots, K \; and \; j = 1, \ldots, N)$ denote student j's knowledge of concept k ($\mathbf{C}$ is the matrix version of C_{kj}). $\mathbf{M}$ is an $Q \times N$ matrix reflecting the intrinsic question difficulty. It is assumed that $K \ll Q, N$ so $\mathbf{W}$ becomes a tall, narrow $Q \times K$ matrix and $\mathbf{C}$ a short, wide $K \times N$ matrix. The model for the binary valued observations $Y_{ij} \in \{0, 1\}$ is then expressed as

$$Z_{ij} = \mathbf{w}_i^T \mathbf{c}_j + \mu_i \quad \forall \quad i = 1, \ldots, Q \;\; j = 1, \ldots, N,$$
$$Y_{ij} \sim Ber(\Phi(Z_{ij})),$$

where $\mathbf{w}_i$ and $\mathbf{c}_j$ denote the ith row and jth column of $\mathbf{W}$ and $\mathbf{C}$, respectively, μ_i is the difficulty level of question i, with $Ber(p)$ denoting a Bernoulli distribution with success probability p and $\Phi(x)$ defined as

$$\Phi(x) = \frac{1}{1 + e^{-x}}.$$

In order to estimate $\mathbf{W}$, $\mathbf{C}$ and μ, Likelihood Maximization (LM) of the observed data Ω_{obs} is performed,

$$(P1): \max_{\mathbf{W},\mathbf{C}} \sum_{(i,j)\in\Omega_{obs}} log(p(Y_{ij}|\mathbf{w}_i, \mathbf{c}_j))$$
$$s.t.$$
$$\|\mathbf{w}_i\|_0 \leq \delta \; \forall \; i \tag{5.1}$$
$$\|\mathbf{w}_i\|_2 \leq \beta \; \forall \; i \tag{5.2}$$
$$\|\mathbf{C}\|_F = \xi \tag{5.3}$$
$$W_{ik} \geq 0 \; \forall \; i, k, \tag{5.4}$$

where $p(Y_{ij}|\mathbf{w}_i, \mathbf{c}_j) = \Phi(\mathbf{w}_i^T \mathbf{c}_j)^{Y_{ij}} [1 - \Phi(\mathbf{w}_i^T \mathbf{c}_j)]^{1-Y_{ij}}$. The Constraint Sets (5.1) through (5.4) guarantee that the solution to $(P1)$ satisfies the identifiability property of matrix factorization. Constraints (5.1) impose the sparsity restriction on matrix $\mathbf{W}$, $\|a\|_0$ counts the number of non-zero entries in vector a and $\|\mathbf{C}\|_F$ denotes the Frobenius norm. Since the optimization under Constraint Set (5.1) requires a combinatorial search, this set of constraints can be replaced by the following sets of constraints (l-1 norm):

$$\|\mathbf{w}_i\|_1 \leq \delta, \; \forall \, i.$$

The constrained optimization problem ($P1$) can be transformed into an unconstrained optimization problem using Lagrange multipliers:

$$(P2): \quad \min_{\mathbf{W},\mathbf{C},W_{ik}\geq 0\ \forall i,k} \quad \sum_{(i,j)\in\Omega_{obs}} -log(p(Y_{ij}|\mathbf{w}_i,\mathbf{c}_j)) + \lambda\sum_i \|\mathbf{w}_i\|_1 + \frac{\mu}{2}\sum_i \|\mathbf{w}_i\|_2^2 + \frac{\gamma}{2}\|\mathbf{C}\|_F^2. \tag{5.5}$$

The resulting optimization problem ($P2$) is biconvex in $\mathbf{W}$ and $\mathbf{C}$ (Lan et al., 2014). *Lan et al.* presented SPARFA-M algorithm for solving this problem (Lan et al., 2014). The idea of that algorithm is based on an alternating approach that splits the problem into two sub-problems. Each sub-problem is optimaly solved in a cyclic manner at each iteration. In general, the alternating approaches perform well in bi-convex optimization, however, when the number of questions, Q as well as the number of students, N increase, the sequential alternating approaches becomes slow, creating a challenge, and hence, an opportunity for exploring parallel implementations.

We now turn to the topic of algorithm parallelization. In many applications, and in particular in student modeling, parallelization pursues the objective of efficiently utilizing computational resources – CPU and memory Richtárik and Takáč (2016). In general, for large-scale machine learning problems, the classical optimization approaches including Newton, Interior-Point and other gradient-based methods become prohibitive due to high computational effort spent in gradient evaluation at each iteration. Several algorithm alternatives exist that manage to mitigate such efficiency challenges. One of these is Stochastic Gradient Descent (SGD) Zhuang et al. (2013). The idea behind SGD is to select a subset of problem variables at random, and estimate the gradient of the objective function with respect to the chosen variables Wright (2015), and then, based on this (partial) estimate, update all the variables. Another viable alternative to the classical full gradient-based algorithms is Coordinate Descent (CD), where at each iteration, the objective function is minimized only in some variables, while all the others are kept constant Wright (2015). In both cases, instead of solving a complex problem requiring much memory, a sequence of less computationally demanding problems is solved.

CD has been successfully employed to tackle a variety of large-scale machine learning problems including Support Vector Machines applications Hsieh et al. (2008), Entropy Maximization Hsieh et al. (2008), and Non-negative Matrix Factorization Cichocki and Anh-Huy (2009), among others. Different variations of CD are distinguished, with its two key components being the update rules and criteria of the selection of sequences in which variables are to be updated Yu et al. (2012). Randomized CD algorithms is a special case of SG methods where the estimation of a gradient is done with respect to only the randomly selected variables.

Although both SGD and CD are popular, they typically struggle handling large-scale data, available for analysis in recommendation systems and intelligent tutoring systems research. However, the inherent sequential and iterative processes allow for exploiting such computational resources as GPU, multi-core or many-core CPUs. In fact, several parallel SGD and CD approaches have been proposed, although mainly for solving convex, smooth optimization problems Sallinen, Satish, Smelyanskiy, Sury, and Ré (Sallinen et al.); Bradley et al. (2011).

Among the existing parallel-SGD methods, particularly among those used for matrix factorization, some are designed for shared-memory systems where all the variables stored in memory are global, i.e., accessible by all the cores Zhuang et al. (2013). For instance, Mini-Batching enables parallelization across samples: here, SGD updates are done by separately working with samples of the data – averaging the gradients obtained for all the samples Cotter et al. (2011). Hogwild Recht et al. (2011) describes an asynchronous, nowadays popular method for parallelizing SGD, which utilizes the "data pass" approach by taking a sample from data and performing an update to the global model without any inter-core communication (for details, see Sallinen, Satish, Smelyanskiy, Sury, and Ré (Sallinen et al.). However, CD-based algorithms have the advantage over SGD-based ones in that the objective function is guaranteed to decrease at every iteration Wright (2015). Moreover, in CD-based algorithms, as convergence occurs, the gradient shrinks to zero, while SGD algorithms obtain non-zero gradient estimates even at the optimum Wright (2015).

Parallel CD algorithms versions vary depending on the implementation and application-at-hand; they may be made to work in a synchronous or asynchronous manner Zhuang et al. (2013). Synchronous algorithms divide the computing tasks/operations into smaller subsets that can be executed in parallel on a multi-core machine; however, the processors need to be synchronized frequently across all the cores, to guarantee the consistency of task (re)assignment in each iteration. Asynchronous implementations, typically preferred in practice, relax this requirement. Convergence analysis of asynchronous methods is more complicated than that of synchronous methods Edmunds et al. (2016), but once properly tuned up, these methods excel in real-world big data analysis applications Wright (2015); Necoara and Clipici (2013); Hsieh and Dhillon (2011); Richtárik and Takáč (2012). In this paper, an asynchronous parallel coordinate descent algorithm using shared memory is presented that can be used to accelerate nontrivial matrix factorization problems in general and SPARFA in application to student modeling in particular.

5.3 The Parallel Coordinate Descent Algorithm

The dimensionality of the students' performance prediction problem increases quickly with the number of students and questions in online courses. However, sequential algorithms become slow and stall even for trivial non-constrained convex optimization problems. To scale SPARFA to larger problems, a parallel version of Coordinate Descent algorithm is implemented. Since many variants of CD are possible, selecting the variant of the sequential CD is the first step in designing PCD. As discussed in detail in Wright (2015), the problem variables to be updated can be selected in a cyclic manner, or alternatively, they can be selected randomly at each iteration.

Certain designs of CD require the problem's objective function to satisfy such proprieties as smoothness and convexity. The optimization problem for SPARFA, introduced in (5.5), is both non-convex and non-smooth (because the regularization function $\|\mathbf{w}_i\|_1 \; \forall \, i = \{1, \ldots, N\}$ is non-smooth), and its objective function is block-separable. In the sequential CD, at each iteration t, a subproblem is created by making a linear approximation to (5.5) along the selected coordinate direction at each iteration. In this implementation, two coordinates, w_{ik} and c_{kj}, are selected at each iteration randomly, and then, the corresponding

subproblem is solved. Algorithm 3 details the randomized coordinate descent algorithm for SPARFA (it is a sequential algorithm).

Algorithm 3 Coordinate Descent Algorithm

Input: Matrix $\mathbf{Y}_{Q\times N}$, K (number of hidden variables, $\lambda, \mu, \gamma, \beta$.
Output: Matrices $\mathbf{W}^*_{Q\times K}$ and $\mathbf{C}^*_{K\times N}$.

1. Initialize randomly $\mathbf{W}$ and $\mathbf{C}$.
2. **while** (stopping criteria) **do**
3. randomly select $i \in \{1,\ldots,Q\}$, $j \in \{1,\ldots,N\}$ and $k \in \{1,\ldots,K\}$
4. calculate $\nabla F_{W_{ik}}$ and $\nabla F_{C_{kj}}$
5. update $W_{ik}^{t+1} \leftarrow \max\{0, W_{ik}^{t+1} - \beta\nabla F_{W_{ik}}\}$
6. update $C_{kj}^{t+1} \leftarrow \frac{1}{1+\beta\gamma}\left(C_{kj}^{-}\beta\nabla F_{C_{kj}}\right)$
7. calculate Objective Function using (5.5)
8. update $\beta \leftarrow \frac{t}{t+2}$
9. $t \leftarrow t+1$
10. **end while**

Selection of step sizes is a crucial element of the algorithm's design that affects the convergence rate; the step size selection rules may or may not be problem-specific. One approach is to select a large value for the step-size (close to one) and adaptively shrink it at each iteration.

5.3.1 Variable Selection Strategy

In Parallel Coordinate Descent algorithm, P coordinates are selected randomly and independently from each other, to be updated in parallel, with P set equal to the number of available threads or cores. The simplest version of PCD in the one where each coordinate, W_{ik}, $i \in \{1,\ldots,N\}$, $k \in \{1,\ldots,K\}$ has an equal chance of being selected, independently from the selection done at previous iterations – one can think of this strategy as a sampling with replacement. For convex objective functions, one can prove the algorithm's convergence under the random selection strategy. However, a better approach – the one used in this paper – is to do sampling without replacement. This scheme provides an opportunity for all the coordinates (decision variables) to be updated in sync. Though the sampling without replacement is computationally more expensive, it achieves a desirable balances between the exploration (of the solution space) and exploitation (of the computational resources). Let Ω_t denote a set of variables that have not been selected up till iteration t, with $\|\Omega_t\|$ denoting the cardinality of set Ω_t. The probability of selecting a variable is then given by $\frac{1}{\|\Omega_t\|}$. Once Ω_t becomes empty, it is re-populated again with all the coordinates.

5.3.2 Variable Updating Strategy

At each iteration of PCD, p pairs of variables selected from $\mathbf{W}$ and $\mathbf{C}$ are being updated. The update for variable W_{ik}, $i \in \{1, \ldots, N\}$, $k \in \{1, \ldots, K\}$, is executed as follows:

$$W_{ik}^{t+1} = \max\{0, W_{ik}^{t+1} - \beta \nabla F_{W_{ik}}\}, \tag{5.6}$$

where $\nabla F_{W_{ik}}$ is the gradient of the objective function in $(P2)$ with respect to variable W_{ik},

$$\nabla F_{W_{ik}} = -\sum_{j=1}^{N} C_{kj}^{t} \Big(Y_{ij} - \frac{1}{1 + e^{-\sum_{k=1}^{K} W_{ik}^{t} C_{kj}^{t}}} \Big) + \mu \frac{W_{ik}^{t}}{\|\mathbf{W}\|_F}.$$

Equation (5.6) guarantees that W_{ik} remains non-negative after every update. This is a projection operator that helps to deal with non-smoothness of ℓ_1-regularization. Similarly, the gradient of the likelihood function with respect to C_{kj}, $\nabla F_{C_{kj}}$, is expressed as

$$\nabla F_{C_{kj}} = -\sum_{i=1}^{Q} W_{ik}^{t} \Big(Y_{ij} - \frac{1}{1 + e^{-\sum_{k=1}^{K} W_{ik}^{t} C_{kj}^{t}}} \Big) + \gamma \frac{C_{ik}^{t}}{\|\mathbf{C}\|_F}. \tag{5.7}$$

The following projection operator enables a faster convergence and limits the growth of C_{kj},

$$C_{kj}^{t+1} = \frac{1}{1 + \beta\gamma} \Big(C_{kj}^{-} \beta \nabla F_{C_{kj}} \Big). \tag{5.8}$$

The PCD algorithm is summarized in the form of a pseudo code in Algorithm 4.

5.4 Computational Results

This section explores the performance of PCD and CD on small-, medium- and large-sized instances of problem $(P2)$. In order to evaluate the performance of PCD, the presented algorithm is tested on 12 synthetically generated data sets. The synthetic data generation for student modeling is not straightforward, in general, since one needs to come up with a realistic model incorporating the differences between the students' abilities and questions' difficulty levels. In this paper, however, it is only natural to use the original SPARFA model, because the object of our analysis is algorithm scalability. Consistent with the SPARFA model assumptions, $\mathbf{W}$ is taken as a non-negative sparse matrix and $\mathbf{C}$ as an arbitrary matrix with bounded elements. The probability that student j answers question i correctly is computed per formula (5.2). The difficulty of questions is assumed to be normally distributed with mean μ_D and σ_D.

5.4.1 Experimental Setup

In order to evaluate the performance of the PCD algorithm, multiple synthetic data sets of different sizes were generated (the corresponding optimization problems have $(Q + N) \times K$ decision variables). The resulting problem instances are categorized into three classes: (1) Small-sized problems, (2) Medium-sized problems, and (3) Large-sized problems. This

Algorithm 4 Parallel Coordinate Descent Algorithm

Input: Matrix $\mathbf{Y}_{Q\times N}$, K (number of hidden variables, P (the number of processes), $\lambda, \mu, \gamma, \beta$.

Output: Matrices $\mathbf{W}^*_{Q\times K}$ and $\mathbf{C}^*_{K\times N}$.

1. Initialize randomly $\mathbf{W}$ and $\mathbf{C}$.
2. Calculate Objective Function using (5.5)
3. **while** (stopping criteria) **do**
4. **In Parallel** on P processes
5. randomly select $i \in \{1,\dots,Q\}$, $j \in \{1,\dots,N\}$ and $k \in \{1,\dots,K\}$ with probability $\frac{1}{\|\Omega_t\|}$
6. **In Parallel** on P processes
7. calculate $\nabla F_{W_{ik}}$ and $\nabla F_{C_{kj}}$
8. update $W^{t+1}_{ik} \leftarrow \max\{0, W^{t+1}_{ik} - \beta\nabla F_{W_{ik}}\}$
9. update $C^{t+1}_{kj} \leftarrow \frac{1}{1+\beta\gamma}\left(C^{-}_{kj}\beta\nabla F_{C_{kj}}\right)$
10. **In Parallel** on P processes
11. Calculate Objective Function using (5.5)
12. update $\beta \leftarrow \frac{t}{t+2}$
13. update $\|\Omega_t\|$
14. $t \leftarrow t+1$
15. **end while**

distinction is due to the fact that the PCD performance in terms of the delivered speedup and convergence changes with an increasing problem size. The largest problem instance generated featured 25,000 students and 600,000 questions (its complexity is equal to that of any other instance with the same number of variables, e.g., one with 600,000 students and 25,000 questions). This instance is large enough to test the scalability of the presented Parallelized SPARFA to any realistic setting; indeed, no MOOC generates as much data. Table 5.1 overviews the designed experimental setup with the 12 data sets.

Table 5.1: Experiments setup for testing Parallel Coordinate Descent algorithm.

Category	Experiment ID	Q	N	K	Variables	CD Iterations	Num of Run	Time Limit (s)
Small	1	5	4	2	18	2000	30	0.1
	2	50	20	4	280	2000	30	0.5
	3	500	200	6	4,200	2000	30	5
	4	1,000	300	6	7,800	2000	30	20
Medium	5	5,000	1,000	8	48,000	1500	25	50
	6	20,000	5,000	8	200,000	1200	25	100
	7	30,000	8,000	9	342,000	1000	25	150
	8	50,000	10,000	9	540,000	500	25	200
Large	9	100,000	12,000	10	1,120,000	500	20	500
	10	250,000	15,000	10	2,650,000	500	20	1000
	11	500,000	20,000	10	5,200,000	200	20	1500
	12	600,000	25,000	12	7,500,000	200	20	2000

5.4.2 Implementation

Some notes regarding the implementation of the PCD are summarized next.

Software: We used a custom code written in C++ with OpenMP, and compiled it with the Intel® C++ Compiler 16.0 with -o3 optimization and -std=c++11. Double precision values were used for matrices $\mathbf{W}$ and $\mathbf{C}$. The observation matrix $\mathbf{Y}$ is binary and highly sparse, so the compressed column and row representations were employed.

Hardware: Different hardware was used for difference SPRAFA problem instances. For experiments 1 through 4, we used a four-sockect Intel(R) Xeon(R) CPU E7- 4830 @ 2.13GHz with 8 cores per socket and 256 GB memory. For experiments 5 through 10, Dell machines with 12x2.40GHz Intel Xeon E5645 Processor Cores and 12 cores per node with 48 GB memory and Linux (RedHat Enterprise Linux 6.1 2.6.32 Kernel) operating systems were used. For the rest of the experiments, we used High Memory Compute (AMD CPUs) with 2.20GHz Processors, 32 cores per node (8 nodes), and 256 GB RAM.

Datasets: The PCD algorithm is run on synthetic data. To generate synthetic data, we generate sparse matrix $\mathbf{W}$ and non-sparse matrix $\mathbf{C}$. Based on the model introduced in Section 2, the observation matrix $\mathbf{Y}$ is generated. Since the observation matrix is highly sparse, a Markov process (namely, random walk) was employed to remove many of its elements to achieve the sparsity of $\mathbf{Y}$. Taking $\mathbf{Y}$ as the data, $\mathbf{W}^*_{Q\times K}$ and $\mathbf{C}^*_{K\times N}$ are then inferred by the PCD algorithm.

Reporting: The main challenge in dealing with high dimension non-convex optimization problems is a large number of local optima, causing the majority of iteration-based algorithms

to get stuck without converging to a global optimum. Another issue with such large-scale optimization problems is that the true optimum is usually unknown in real-world applications. Consequently, defining a stopping criterion is problematic. In this paper, we consider two stopping criteria, one based on runtime and the other based on the number of executed iterations. According to the first criterion, the PCD algorithm traverses the solution space till a pre-set given time, at which point the current best solution is returned and its objective function is reported; this objective function, defined in (5.5), is the negative log-likelihood function that includes regularization terms. According to the second criterion, the PCD executes a pre-selected fixed number of iterations and then returns the best found solution. To ensure that the randomness in the initial conditions does not have a significant effect on the solution quality, each experiment is run several times as reported in Table 1. In addition, the time from the algorithm's initialization till its termination is noted and reported. The speedup (S_p) and efficiency (E_p) are also reported for each experiment. The speedup is defined

$$S_p = \frac{T_s}{T_p}, \tag{5.9}$$

where T_s is the execution time of the sequential algorithm and T_p is the execution time of the parallel algorithm with p processes. Note that the ideal (best possible) speedup is $S_p = p$, referred to as a linear speedup. The efficiency is defined as

$$E_p = \frac{S_p}{p}. \tag{5.10}$$

5.4.3 Results for Small-sized Problem Instances

The first set of the experiments had four instances with 18, 280, 4,200 and 7,800 variables, respectively. The PCD performance as a function of the number of cores was evaluated under both stopping criteria. Figure 5.1 illustrates how the PCD algorithm convergence times grow as the number of coordinates (i.e., cores) increases. It shows that when more coordinates are updated at once, the algorithm converges faster, particularly as the problem size increases. Figure 5.2 compares the averages of the objective function obtained from 30 runs for each small-sized problem instance. The results show that by increasing the number of cores (i.e. threads generated in OpenMP), the performance of PCD improves, especially if a time limit (e.g. two minutes) is considered as the stopping criterion. To explore how the addition of computational resources affects the run time, particularly when the algorithm is run with a fixed number of iterations, the speedup values are reported in Figure (5.3). For the small-sizes problem instances, the overhead cost of inter-core communication turns out to be high to reap the substantial benefits from running the CD algorithm in parallel. This phenomenon is particularly true for synchronous CD because of an unbalanced load distribution over the cores. Table 5.2 summarizes the speedup and efficiency values obtained by running the experiments on a different number of cores. Note that the value of T_p, used to calculate S_p, is taken as the average over 30 runs.

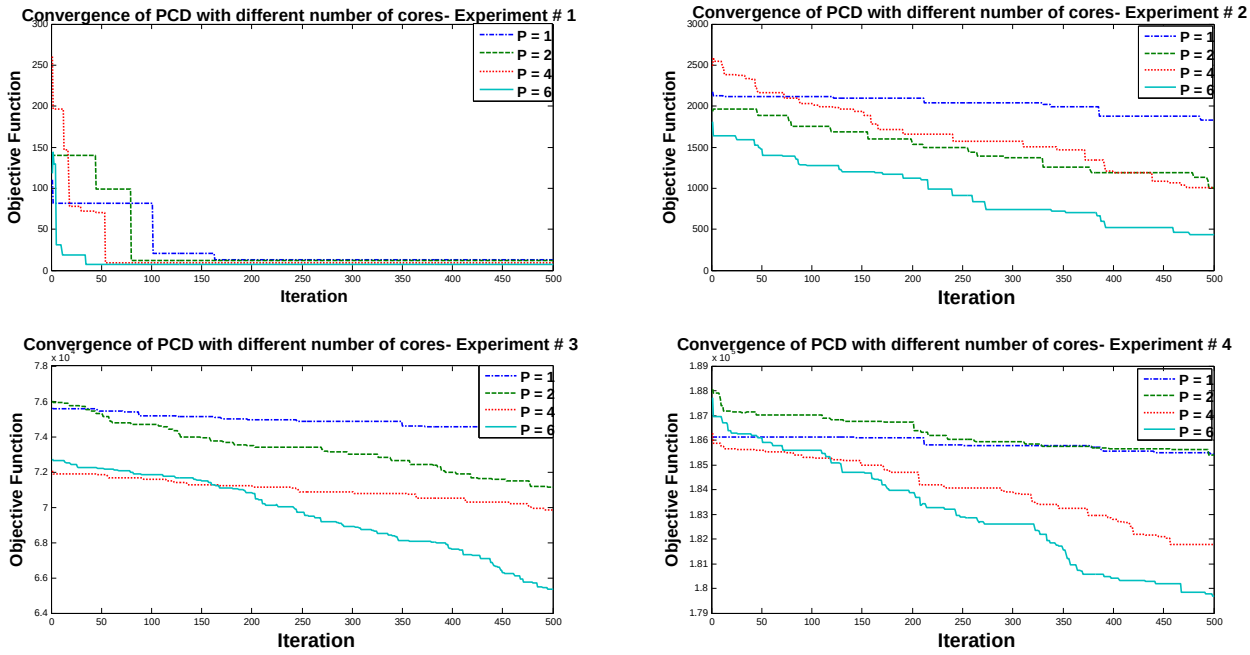

Figure 5.1: Comparison of the PCD convergence rates achieved with a different number of cores (p=1,2,4 and 6). Using more cores results in a faster convergence.

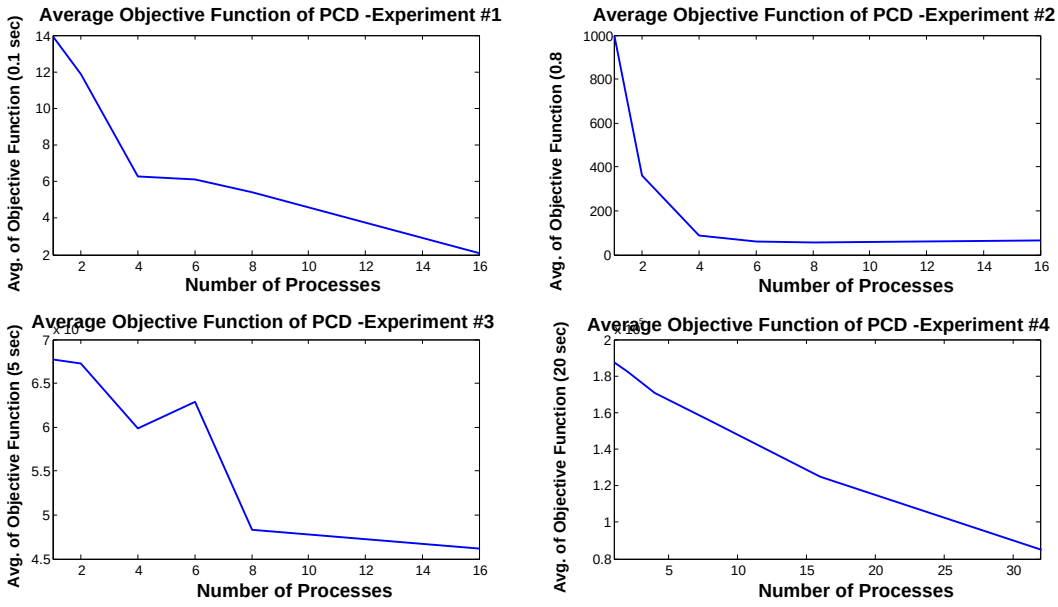

Figure 5.2: The averages of the objective function for the small-sized instances over 30 runs.

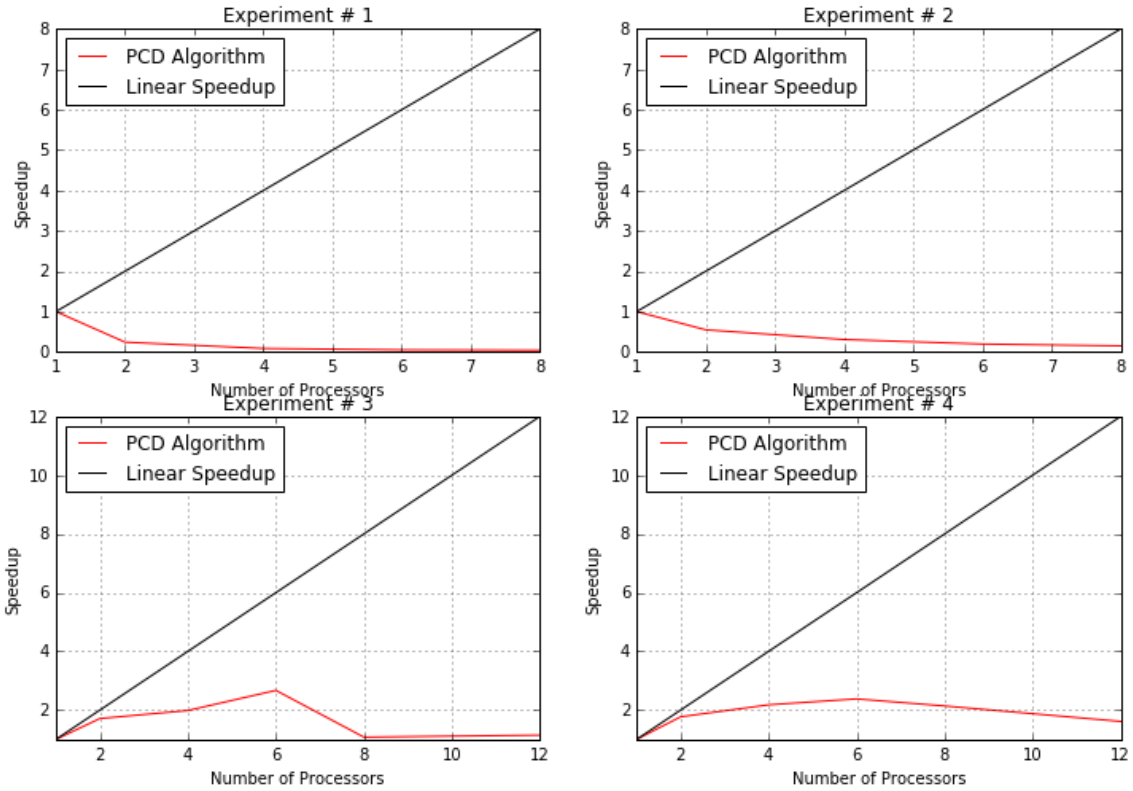

Figure 5.3: The averages of speedup for the small-sized instances.

Table 5.2: Speedup and efficiency for instances 1 through 4 (NR: Not Reported).

Number of Cores	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
	S_p	$E_p(\%)$	S_p	$E_p(\%)$	S_p	$E_p(\%)$	S_p	$E_p(\%)$
1	1	100	1	100	1	100	1	100
2	0.242	12.1	0.548	27.4	1.711	85.6	1.770	88.5
4	0.086	2.2	0.307	7.7	1.984	49.6	1.778	44.5
6	0.052	0.9	0.196	3.3	2.670	44.5	1.880	31.3
8	0.043	0.5	0.150	1.9	1.076	13.4	2.839	35.5
12	NR	NR	NR	NR	1.145	9.5	1.613	13.4

5.4.4 Results for Medium-sized Problem Instances

The medium-sized experiments include four instances with 48,000 to 540,000 variables. The performance of PCD algorithm was tested on these instances with a different number of cores. Figure 5.4 depicts the averages of the objective function over 25 runs for four instances with random initialization. Adding more cores results in improving the average of the objective function. As discussed before, the speedup defined in (5.9) indicates the impact of parallelization on runtimes. The speedup averages for the medium-sized instances are

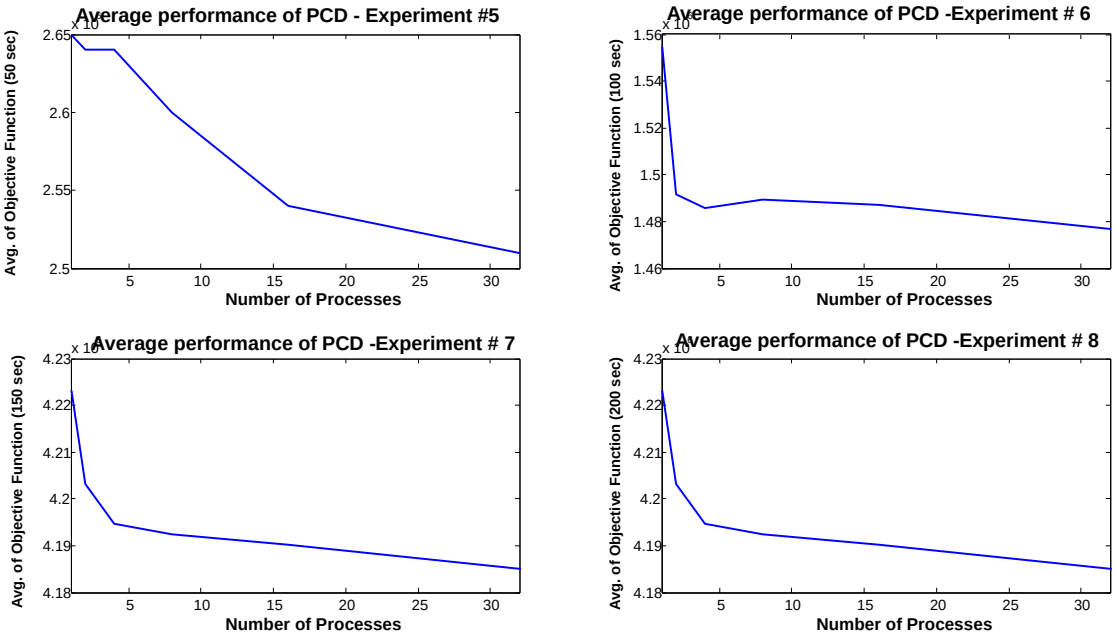

Figure 5.4: The averages of the objective function for the medium-sized inctances improve over 25 runs.

compared with respect to a linear speedup (the ideal speedup) in Figure 5.5. The speedup

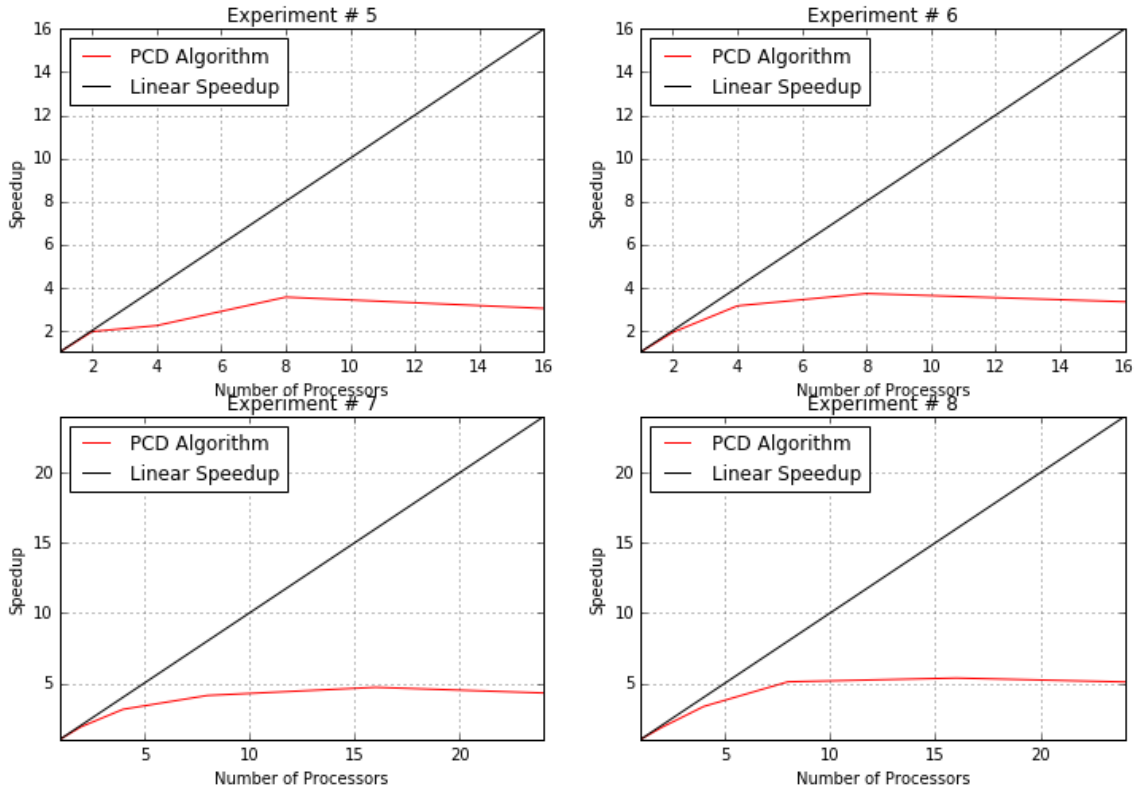

Figure 5.5: The averages of speedup for the medium-sized instances.

values obtained for medium-sized problem instances lie below the line for the linear speedup. However, the speedup improves with a growing problem size. For instances 5 and 6, the maximum speedup is reached with eight cores while using 16 cores gives the maximum speedup in instances 7 and 8. Table 5.3 reports the speedup and efficiency values from running the PCD algorithm on instances 5 to 8 with a different number of cores. In the larger instances, adding more computational resources (i.e. cores) improves the speedup.

Table 5.3: Speedup and Efficiency for experiments 5 through 8 (NR: Not Reported).

Number of Cores	Experiment 5		Experiment 6		Experiment 7		Experiment 8	
	S_p	$E_p(\%)$	S_p	$E_p(\%)$	S_p	$E_p(\%)$	S_p	$E_p(\%)$
1	1	100	1	100	1	100	1	100
2	1.947	97.4	1.913	95.7	1.894	94.7	1.854	92.7
4	2.223	55.6	3.142	78.6	3.143	78.6	3.348	83.7
8	3.546	44.3	3.712	46.4	4.128	51.6	5.103	63.8
16	3.023	18.9	3.333	20.8	4.711	29.4	5.370	33.6
24	NR	NR	NR	NR	4.305	17.9	5.092	21.2

5.4.5 Results for Large-sized Problem Instances

The large-sized experiments include the instances with more than one million variables. Figure 5.6 illustrates the averages of the SPARFA objective function with a different number of processes (cores) used exploited for the gradient-driven updates. Again, adding more cores improves the objective function value, on average, up to a certain threshold. Figure 5.7 shows the speedup values and the level of parallelism that PCD achieves over large-sized problem instances. In this set of experiments, the achieved speedup is greater than 7x. This level of speedup indicates that running the PCD algorithm on a large optimization problem, which would otherwise take a week to solve, will take just one day. Table 5.4 presents the achieved speedup and efficiency values for the large-sized instances. Running the PCD algorithm on eight cores results in more than 70% efficiency. Note that any larger efficiency in a shared memory system is unlikely, due to the overhead computing costs, unless one uses all cache levels (i.e., L1, L2, L3, etc.) in a very efficient way.

An additional study is performed to establish the value of the parallel implementation of SPARFA over its non-parallel counterpart. To this end, consider a practical scenario where SPARFA analysis is to be performed regularly, within a limited time window. In a real-world setting, one can assume that the inference results are to be updated overnight as students take on more questions, and hence, more data become available. Then, any SPARFA run or runs should not take more than eight hours. Indeed, taking into account the fact that the model contains a non-convex optimization component, it is highly recommended to run the optimization algorithm multiple times, with randomly selected values of $\mathbf{W}$ and $\mathbf{C}$. To conduct an analysis for this case, 15 experiments including 5, 10 and 20 sets of 72-, 36- and 18-minute runs, respectively, were performed, with each set taking exactly 8 hours in total (e.g., 20×18 minutes $= 8$ hours), using 1, 4, 8, 16 and 32 cores.

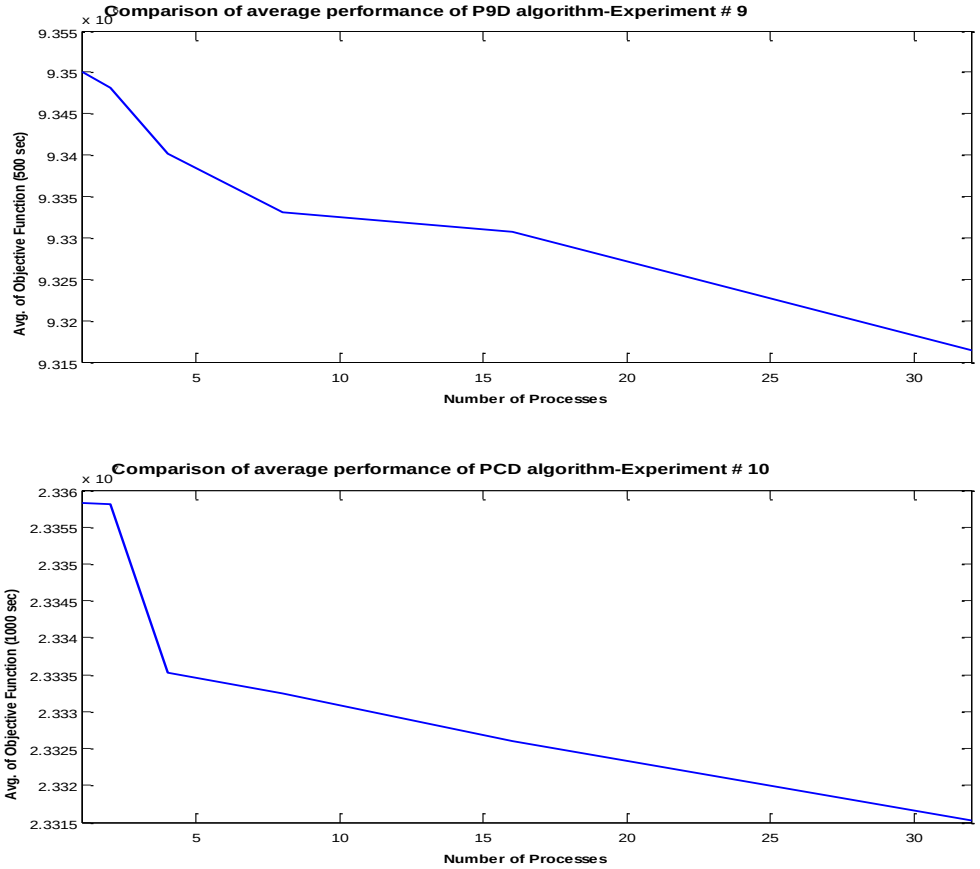

Figure 5.6: The averages of objective function for the large-sized instances over 25 runs.

Table 5.4: Speedup and efficiency of experiments 9 to 12.

Number of Cores	Experiment 9		Experiment 10		Experiment 11		Experiment 12	
	S_p	$E_p(\%)$	S_p	$E_p(\%)$	S_p	$E_p(\%)$	S_p	$E_p(\%)$
1	1	100	1	100	1	100	1	100
2	1.951	97.6	1.973	98.6	1.992	99.6	1.947	97.4
4	3.578	89.4	3.645	91.1	3.421	85.5	3.506	87.7
8	5.424	67.8	5.710	71.4	5.194	64.9	5.787	72.3
16	6.221	38.9	6.838	42.7	7.155	44.7	6.995	43.7
24	5.763	24.0	6.326	26.4	7.247	30.2	7.403	30.8
32	4.462	13.9	5.779	18.1	6.998	21.9	7.343	22.9

Figure 5.8 shows the results for both the sequential and parallel CD algorithms on the largest problems instance. The sequential SPARFA algorithm (CD with one core) is compared against its parallel versions utilizing 4, 8, 16 and 32 cores. Because in any run, the algorithm might get stuck in a local minimum, it was run with multiple restarts performed within an 8-hour time slot, and the average and the best obtained objective function value (calculated using (5.5)) was reported. The results show that the parallel SPARFA runs return

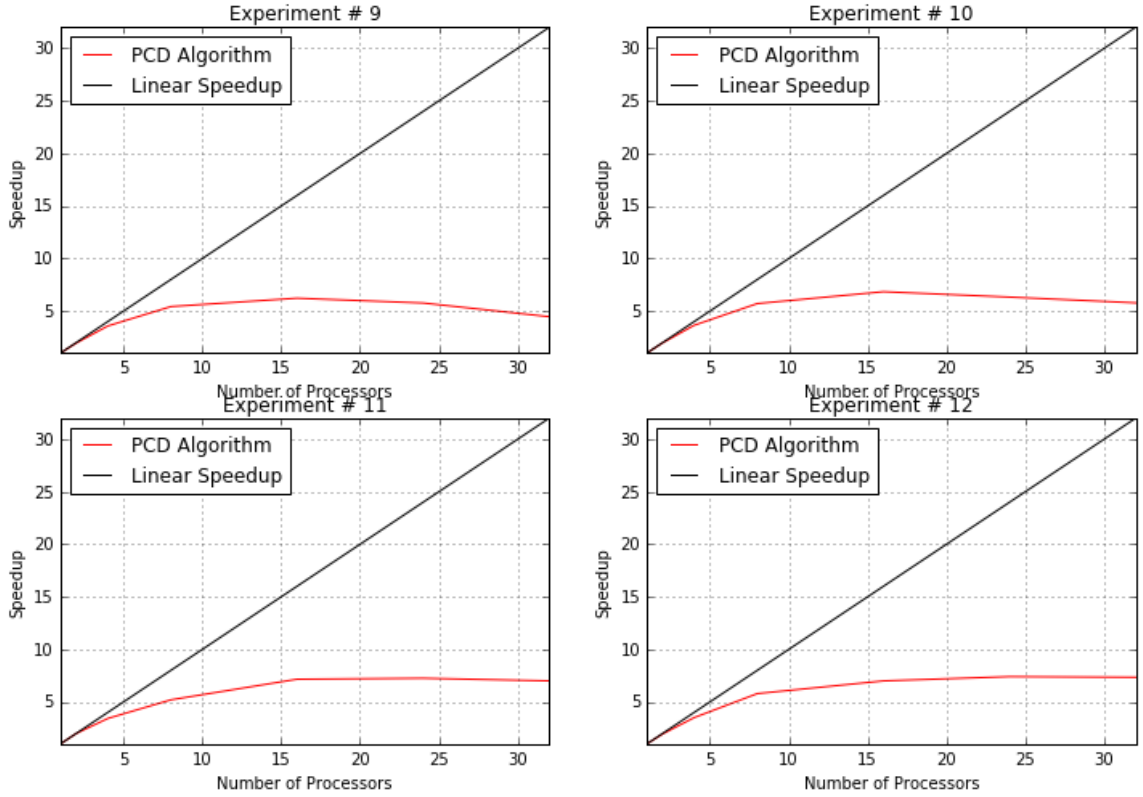

Figure 5.7: The average speedup of large-size problems.

significantly better objective function values. This observation indicates that parallelizing SPARFA is a worthy undertaking

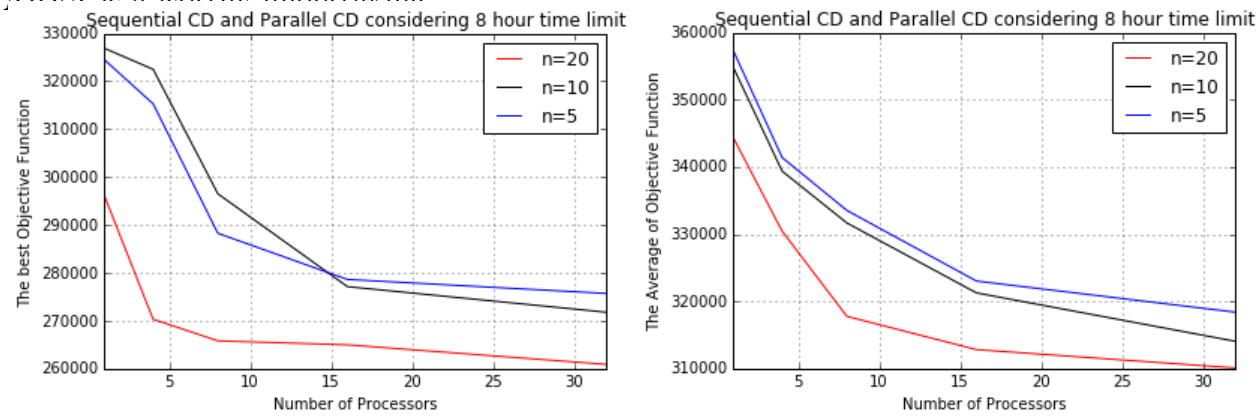

Figure 5.8: The best objective function (left) and average of objective function (right) obtained in an 8-hour run with 5, 10 and 20 random initialization utilizing 1,4, 8, 16 and 32 cores.

In addition, the performance of the parallel SPARFA (with 32 cores) is compared against its sequential counterpart, iteration by iteration. Since the performance of both sequential and parallel algorithms rely upon starting points, this analysis tracks the improvement achieved during the same number of iterations in the runs with the same initial solutions. In this experiment, each algorithm is given 18 minutes (which amounts to 34 iterations per run)

to optimize the objective function of the LM problem, for ten randomly selected initial solutions. This setting was selected because multiple 18-minute runs produced the best overall inference results over an 8-hour time period in the previous set of experiments. Figure 5.9 depicts the observed objective function value dynamics over time. Indeed, it confirms that the parallel SPARFA consistently achieves lower objective function values, offering a 5-10% improvement; this improvement is substantial even with multiple algorithm runs. Figure 5.9 (right) details the improvement observed in LM optimization achieved with the best initial solution.

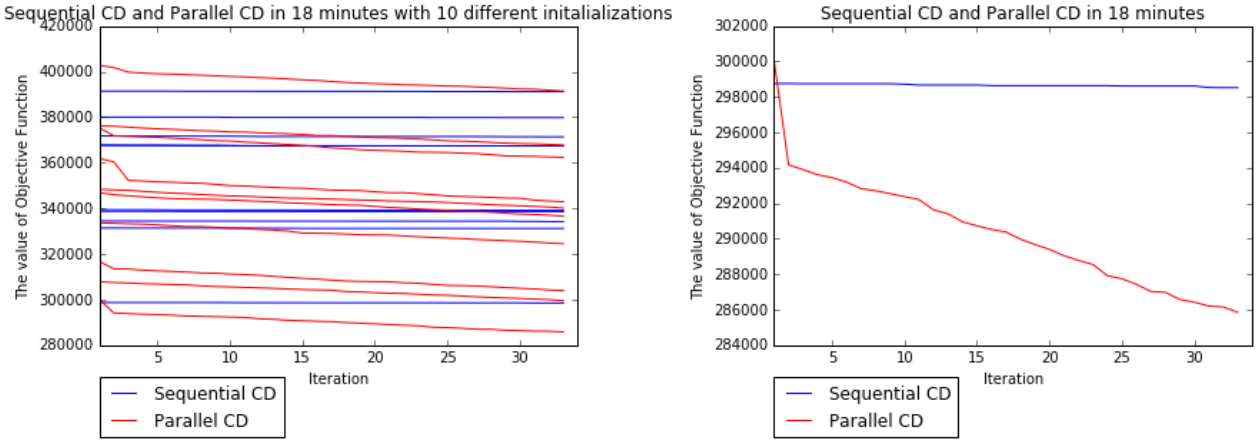

Figure 5.9: Parallel and Sequential SPARFA in 10 runs with randomly initialized $\mathbf{C}$ and $\mathbf{W}$ (left) Comparing parallel and sequential SPARFA with the best initial candidate.

5.5 Conclusion

This paper presents a parallel algorithm for SPARFA – a machine learning based approach to student modeling that allows for predicting students' performance on a set of questions they have not yet taken. To this end, a parameteric model is fit to the available data of the observed student question-answering outcomes using MLE. This MLE task results in a non-convex, non-smooth optimization problem, which the originally proposed algorithm solves using an alternating gradient approach, with two sub-problems solved iteratively till the convergence of both.

This paper attacked the scalability challenge in SPARFA, and offered a parallel version of CD algorithm to solve it; the algorithm runs seven times faster, on average, than the sequential approach, with higher speedups observed on larger problem instances. At each iteration, the algorithm selects p variables without replacement to be updated in parallel in a shared memory setting. The performance of the PCD algorithm was tested on several synthetic data sets, with the largest one featuring hundreds of thousands of students and questions, thus being representative of any realistically large MOOC.

The experimental results confirm that the parallelization significantly increases the efficiency of use of computational resources (i.e., more than 70% efficiency with running the PCD

algorithms on eight cores in parallel). Note that due to inter-core communication, adding more computational resources beyond some threshold does not lead to further speedup or efficiency improvements; the threshold depends on the problem size.

Chapter 6

Predictive Student Modeling

6.1 Introduction

Machine learning techniques have been extensively applied in the education domain since the first conference of educational data mining in 2008 (Thai-Nghe et al., 2009). The techniques such as Logistic Regression, Decision Trees, Bayesian Networks, etc. address various teaching/learning-challenges questions including student performance prediction, grouping students by ability, and directing students to appropriate learning materials and courses (Minaei-Bidgoli et al., 2003; Bekele and Menzel, 2005; Kotsiantis and Pintelas, 2005).

Student performance prediction is one of the most actively researched educational data mining tasks addressed with the help of machine learning tools (Toscher and Jahrer, 2010; Cetintas et al., 2010; Yu et al., 2010). The goal of student performance prediction lies in evaluating how much knowledge students have and how they can be expected to perform on given assessment tasks. Importantly, since knowledge levels and skills tend to improve with time, the sequence in which students take on the assessment tasks is an important determinant in the inference problem formulations and outputs (Thai-Nghe et al., 2011). The KDD Cup 2010 featured student performance prediction as its problem of choice, challenging scientists to infer how students learn and adapt to new problems, and what knowledge levels/dimensions are required for correctly solving a given problem/question, based on observed student performance data. Improving student models to predict students performance can save millions of hours of students' time and effort (Cen et al., 2006). Developing powerful predictive models can also improve, or even lead to changing the format of, the current standardized tests such as SAT and GRE. All these efforts are primed to ameliorate the pressure and stress on students (Feng et al., 2009; Thai-Nghe et al., 2011).

The number of efforts undertaken to improve student performance prediction demonstrate the importance of modeling in educational systems research. However, most of the existing modeling studies are based on traditional classification/regression techniques, which fail to deal with the sparsity of observed data. Some very recent studies, however, report on the successes achieved through the use of matrix factorization techniques, previously adopted in recommender system research, for predicting students performance (Thai-Nghe et al., 2010; Toscher and Jahrer, 2010; Thai-Nghe et al., 2011; Thai-Nghe, 2011; Lan et al., 2014). Although learning and problem-solving are multifaceted activities, which are based

on complex cognitive processes fundamentally different from consumer logic, the collaborative filtering models were found capable of inferring latent dimensions of knowledge and explaining students' performance. With little observational data available, exploiting matrix factorization methods turns out particularly useful (Thai-Nghe et al., 2009, 2011).

It is worthwhile to stress that the main driver behind the efforts toward predicting students performance is not the need to forecast students' future successes of failures on assignments, but the desired ability to identify an optimal way to teach students. Given this objective, a useful student model must account for the fact that learning happens over time – during students' interactions with course content. Moreover, it is desirable to discover *why* and *how* students acquire knowledge, and hence, models with interpretable parameter values are particularly valuable.

This chapter offers a new framework for student modeling, based on constrained tensor factorization. It presents three models that are able to handle highly sparse data and estimate students' conceptual material understanding levels, while recognizing both the temporal dynamics of knowledge acquisition and the differences in students' learning abilities. Two optimization algorithms are designed to deal with the likelihood maximization problems for the estimation of the models' parameters. The obtained computational results showcase a solid performance of the models in prediction tasks, while explicitly revealing the knowledge acquisition dynamics of students, all estimated from observations.

6.2 Student Prediction Models

Student performance prediction has been studied in traditional education systems for a decade and many works have addressed several aspect of this challenging problem. There exist a number of studies on Item Response Theory (IRT), that estimates the probability of response that a student provide to a particular item (Reckase, 2009; Brinton and Chiang, 2015). Cen *et al.* proposed Latent Factor Analysis (LFA) that represents accumulated learning multiple skills for students and the probability of mastering the skills is determined by the knowledge accumulation (Cen et al., 2006). The LFA model is an extension of the Item Response Theory model (Thai-Nghe, 2011). The Performance Factors Analysis (PFA) presented by Pavlik *et al.* is another student learning model that improves LFA. Similar to the LFA, PFA also makes use of a logistic regression model to predict student correctness based on the student's number of prior successes (Pavlik Jr et al., 2009).

More advanced student prediction models mainly rely on matrix factorization methods successfully implemented in recommendation systems (Bell and Koren, 2007; Koren et al., 2009; Paterek, 2007). In other words, predicting student performance is modeled as rating prediction in recommendation systems such that students, questions, and performances are replaced with users, items, and rates, respectively (Thai-Nghe et al., 2010). It has been shown that the matrix factorization techniques work well to predict student performance and excel other methods such as global average, user average, item average, user-item-baseline (Koren, 2010), regularized logistic regressions (Komarek and Moore, 2005) and Bayesian Knowledge Tracing models (d Baker et al., 2008) (see (Thai-Nghe, 2011) for more details).

Thai-Nghe (2011) proposed variants of matrix factorization models that specifically consider latent factor models for student modeling. These proposed models implicitly incor-

porate the student and task latent factors (such as slip and guess). Moreover, the authors employed k-nearest neighbors collaborative filtering to consider similarities of students using the correlations between the students and the tasks. They also improved the student performance prediction by taking temporal/sequential information into account and utilized basic tensor factorization methods to model the student latent factors and the sequential/temporal effects.

More recently, there have been significant studies aimed at the analysis and development of more effective matrix factorization methods (Khajah et al., 2014; Lan et al., 2014b), which seek to more precisely impute missing elements of a matrix given possibly noisy or corrupted observations (Soni et al., 2016). The recently proposed SPARse Factor Analysis (SPARFA) framework (Lan et al., 2014) as a state-of-the-art model that uses probabilistic matrix factorization, excels other factor analysis approaches in prediction students' performance (Lan et al., 2014a). It provides the probability that a students answers a given question or solves a problem correctly based on her understanding of a set of underlying course concepts, on the concepts involved in each question, and each question's intrinsic difficulty (Lan et al., 2014). To consider more specifically the effect of time on students learning, a model based on approximate Kalman filter for sparse factor analysis was proposed that traces learner concept knowledge over time by considering student interaction with learning resources, such as textbook sections, lecture videos or the forgetting effect and estimates the content organization and difficulty of the questions (Lan et al., 2014b). Other machine learning based techniques such as RNN model with sigmoid units and a Long Short Term Memory (LSTM) model called Deep Knowledge Tracing have been proposed to the problem of predicting students' performance to based upon their activity history (Piech et al., 2015).

However, the above mentioned methods are unable to infer student conceptual understanding because the majority of these models including SPARFA and all matrix factorization-based models do not consider the fact that learning takes place overtime and do not pay attention to knowledge accumulation. Although Bayesian Knowledge Tracing model and its extensions d Baker et al. (2008); Yudelson et al. (2013); González-Brenes et al. (2014) as well as more recently proposed methods such as SPARFA-trace Lan et al. (2014b), RRN-based models, etc. work with chronological data, they suffer from aiming at student conceptual learning and knowledge accumulation due to interaction with learning materials. Even an powerful model such as SPARSE-trace, for instance, does not control knowledge acquisition over time so that a learner's knowledge may deteriorate as a result of interaction with learning materials. In addition, it is important that the models provide interpretable parameters so that instructors are equipped to monitor students progress at each point of time. Finally, difference among knowledge components are neglected in most existing student models as well as the fact that students may lead or lag in some of the learning contents and some of them are more creative to answer a given question.

6.3 Sparse Tensor Factorization Models

Matrix and tensor decomposition is at the core of providing machine learning-based solutions in many application area, including recommender systems (Frolov and Oseledets, 2016), feature extraction (Evgeniou and Pontil, 2007), signal processing (Kurucz et al., 2007), social

tagging systems (Rafailidis and Daras, 2013), computer vision and graphics, missing value estimation (Liu et al., 2013) and blind source separation (Grasedyck et al., 2013). In this section, a detailed description of the proposed tensor factorization-based modeling approach, along with some general concepts required to explain the developed models, are provided (for more details of the key mathematical properties of tensor factorization, see (Grasedyck et al., 2013; Comon, 2014; Liu et al., 2013)). The major advantage of tensor-based models, particularly those used in recommendation systems, is their ability to incorporate multiple features, and importantly, time into the traditional user-item interactions (Frolov and Oseledets, 2016). This latter trait becomes most useful when one attempts to make inference from the observations of student performance, under the assumptions that knowledge acquisition and development of skills occur *while* students are being observed.

6.3.1 Preliminaries and Notations

An extension of the concept of "matrix", an N-way *tensor* is a mapping from a linear space to another space with N distinct dimensions. In this paper, tensors, matrices and vectors are denoted with calligraphic capital letters (e.g., $\mathcal{Y} \in R^{I_1,\ldots,I_N}$), bold capital letters (e.g., W) and bold lowercase letters (e.g., $\mathbf{d}$), respectively. For $\mathcal{Y} \in R^{I_1,I_2,I_3}$ of size $I_1 \times I_2 \times I_3$, a compact format $\mathcal{Y} = [\mathcal{Y}_{ijk}]$ is used, where $\mathcal{Y}_{ijk}$ is an element of $\mathcal{Y}$ such that $i \in I_1$, $j \in I_2$ and $k \in I_3$.

Let $\mathcal{Y} \in \{0,1\}^{Q,N,T}$ denote a binary-valued data set of student performance observations of size $Q \times N \times T$, with entry $\mathcal{Y}_{ijt} = 1(0)$ signaling that student $j = 1,\ldots,Q$ answered question $i = 1,\ldots,N$ at time $t = 1,\ldots,T$ correctly(incorrectly); here, Q, N and T are the number of students, the number of questions and the number of time slots (e.g., weeks) for which the data were collected, respectively. The sparsity of tensor $\mathcal{Y}$ depends on the number of available questions Q and the level of activity of each student in taking on some of these questions. It is typical to expect that many questions will be answered only by a small subset of students, resulting in incomplete data. Similar to SPARFA and other ITS model assumptions, it is assumed here that a certain number K of latent knowledge components (KCs) are required to gain mastery of the subject matter that the students are exposed to. It is worth nothing that K is taken to be significantly smaller than Q and N (i.e., $K << N, Q$) (Lan et al., 2014). To acknowledge the fact that the questions may have different intrinsic difficulty, d_i indicates the difficulty level of question $i = 1,\ldots,Q$; higher values of d_i indicate higher difficulty. It is also reasonable to assume that students have different skills, knowledge levels, background and aptitude: θ_j denotes the ability of student $j = 1,\ldots,N$ to learn new material; higher values of θ_j indicates that the student is more likely to answer a question correctly. The vectors $\mathbf{d}$ and $\boldsymbol{\theta}$ contain the difficulty levels of all the questions and the ability levels of all the students, respectively. To be compatible with the notations in Lan et al. (2014), let W_{ki} represent the degree that KC k is involved in question i: $W_{ki} = 0$ means that answering question i does not require KC k. Also, let C_{kj} represent the level that student j achieves in KC k. Finally, let V_{kt} represent the relative level that learners (as a group) tend to acquire in KC k by time t. In summary, matrix $\mathbf{W} = [\mathbf{w_1}\ \mathbf{w_2}\ \ldots\ \mathbf{w_Q}]$ relates questions and KCs, matrix $\mathbf{C} = [\mathbf{c_1}\ \mathbf{c_2}\ \ldots\ \mathbf{c_N}]$ relates students and KCs, and matrix $\mathbf{V} = [\mathbf{v_1}\ \mathbf{v_2}\ \ldots\ \mathbf{v_T}]$ relates KC and time.

6.3.2 Static Student Model (SSM)

As an extension of Matrix Factorization (MF), Tensor Factorization brings the time as very important feature in students' learning into ITSs. In 2D case (i.e., matrix), the "rank" is an important feature to extract some type of global information. Tensor factorization extends the concept of low-rank approximation of a matrix to higher orders (i.e., higher order singular value decomposition). Indeed, the goal of tensor factorization is to build a model that can learn more complex patterns from real observations, $\mathcal{Y}$. In other words, tensor factorization methods are used to reconstruct tensor $\mathcal{T}$ using decomposition techniques such that $\mathcal{Y} \approx \mathcal{T}$. There are several decomposition techniques that can be employed to estimate students performance on unanswered questions over time.

Canonical Polyadic (CP) Decomposition: A straightforward way to extend the concept of SVD to higher order is to add another factor to the low-rank approximation model Frolov and Oseledets (2016). Considering a latent factor model with three dimensions (e.g., students, questions and time), the reconstruction tensor can be obtained as follows:

$$\mathcal{T} = \sum_{k=1}^{K} \psi_k \mathbf{w_k} \otimes \mathbf{c_k} \otimes \mathbf{v_k},$$

where $\otimes$ indicates the outer product of vectors and $\boldsymbol{\psi}$ is a vector of length K with elements $\psi_1 \geq \ldots \geq \psi_K$. However, it is safe to assume that the decomposition relies more on $\mathbf{W}, \mathbf{C}$ and $\mathbf{V}$, therefore, $\boldsymbol{\psi}$ can be safely neglected. Figure 6.1 illustrates the tensor factorization problem using CP Decomposition approach.

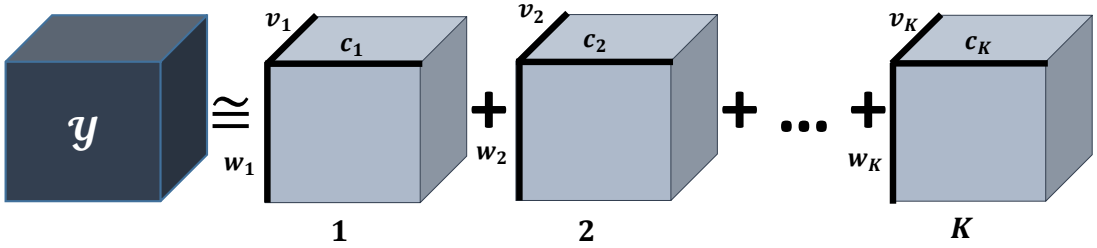

Figure 6.1: Tensor Factorization Model based on CP decomposition.

Tucker Decomposition (TD): Another way of extending SVD to a higher order decomposition is to replace the diagonal matrix Σ, diagonal matrix of eigenvalues in SVD Frolov and Oseledets (2016) with a 3-way tensor called core tensor.

A traditional issue with decomposition models, even 2-D models, is that the decomposition is not unique, however the optimization problem, usually minimizing $\|\mathcal{Y} - \mathcal{T}\|_F$ with respect to multilinear rank is well-defined Frolov and Oseledets (2016). In general, tensor factorization is not well-posed, especially when the tensor is highly sparse Lan et al. (2014b). Similar to SPARFA, our first key assumption that motivates a well-posed solution is the fact that only a small number of knowledge concepts are involved in each course ($K << Q, N, T$).

Our approach to model this problem is based on a statistical inference that enables one to define a probability distribution over each learner's performance such that what the probability that learner j will answer question i at time t correctly considering five factors:

(i) the learners knowledge of a set of latent, abstract concepts, (ii) how the question is related to each concept, and (iii) the intrinsic difficulty of the question, (iv) the dynamic of learning and knowledge growth overtime and (v) students have different learning characteristics, level of smartness and effort.

Given these assumptions, a model with interpretable parameters is needed to predict the performance of student j on question i at time t, with 1 representing a correct response and 0 an incorrect response. In order to deal with the underlying uncertainty of the problem due to student complex behaviors, one can construct a probabilistic model that provides the probability of correctly answering question i at time t by student j correctly. In this model, the difficulty of questions and the student parameter are included as well. Let $\mathcal{T}$ denote the reconstructed tensor obtained as follows:

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ki} C_{kj} V_{kt} - d_i + \theta_j \quad \forall \quad i = 1, \ldots, Q \;\; j = 1, \ldots, N, \; t = 1 \ldots T, \tag{6.1}$$

where d_i, and θ_j denote the difficulty of question i and student j parameter, respectively. The probability that student j solves question i at time t follows a Bernoulli distribution.

$$\mathcal{Y}_{ijt} \sim Ber(\Phi(\mathcal{T}_{ijt})), \tag{6.2}$$

In (6.2), *Ber(p)* shows the Bernoulli distribution with success probability p and $\Phi(x)$ is *logit* function defined as follows:

$$\Phi(x) = \frac{1}{1 + e^{-x}}$$

Given $\mathbf{w_i}$, $\mathbf{c_j}$, $\mathbf{v_t}$, d_i and θ_j, the probability of student j's performance on question i at time t follows the following distribution:

$$P(\mathcal{Y}_{ijt} | \mathbf{w_i}, \mathbf{c_j}, \mathbf{v_t}, d_i, \theta_j) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}} [1 - \Phi(\mathcal{T}_{ijt})]^{1 - \mathcal{Y}_{ijt}} \tag{6.3}$$

Let $\Omega_{obs} \subseteq \{1, \ldots, N\} \times \{1, \ldots, Q\} \times \{1, \ldots, T\}$ indicate the set of observed performances. One can use Maximum Likelihood Estimation (MLE) approach to estimate parameters of (6.3). As discussed in Lan et al. (2013), estimating the parameters of (6.3) is not a well-posed problem due to over-fitting, interpretability and identifiability of matrices $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$. Assumptions below are useful to cope with such problems Lan et al. (2014):

- **Low-rank Property:** In traditional matrix factorization using SVD method, the hidden variable $K = min(M, N)$ is the rank of SVD. However, in practice, truncated SVD where $K < min(M, N)$ provides reasonable approximation of the original matrix. In educational domain, K also represents the number of knowledge components that students learn during the course. In reality, the number of such knowledge components is significantly smaller compared to the number of students and questions.

- **Sparsity:** Creating a question which covers most of the concepts thought in a course is extremely difficult; therefore, one can postulate that $\mathbf{W}$ contains many zeros (it is so-called sparse). This assumption remarkably increases the identifiability of the corresponding MLE problem.

- **Non-negativity:** In general, interpretability is considered as a desired structural property of many machine learning models. As discussed in the previous section, many models in ITSs significantly suffer from the lack of interpretability. Negative values of parameters of this model make the interpretation of models more difficult.

Considering all these assumptions, the MLE optimization model needs a set of constraints to propose a well-defined problem. To meet the sparsity assumption, the number of non-zero elements of $\mathbf{W}$ should be limited such that $\|\mathbf{w_i}\|_0 \leq \delta$ where $\|.\|_0$ counts the number of non-zero entries in the corresponding vector. However, $\|.\|_0$ is not differentiable and converts the MLE problem into a combinatorial optimization problem. Therefore, one can relax this set of constraints to ℓ-1 norm. Although ℓ-1 norm makes the objective function of the optimization problem non-smooth, efficient optimization techniques which will be discussed later can be employed to deal with this problem. ℓ-2 norm of $\mathbf{W}$ prevents the elements of the matrix to grow unboundedly and helps the optimization algorithms to converge Lan et al. (2014). Using Frobenius norm, $\|\mathbf{C}\|_F$ and $\|\mathbf{V}\|_F$ suppress arbitrary scalings between the entries in the matrices.

$$(P1): \quad \max_{\mathbf{W},\mathbf{C},\mathbf{V},\mathbf{d},\boldsymbol{\theta}} \quad \sum_{(i,j,t)\in\Omega_{obs}} log(P(\mathcal{Y}_{ijt}|\mathbf{w_i},\mathbf{c_j},\mathbf{v_t},d_i,\theta_j))$$

$$s.t.$$

$$\|\mathbf{w}_i\|_1 \leq \delta \;\; \forall \;\; i=1,\ldots,Q \tag{6.4}$$

$$\|\mathbf{w}_i\|_2 \leq \beta \;\; \forall \;\; i=1,\ldots,Q \tag{6.5}$$

$$\|\mathbf{C}\|_F = \xi \tag{6.6}$$

$$\|\mathbf{V}\|_F = \eta \tag{6.7}$$

$$W_{ki}, C_{kj}, V_{kt} \geq 0 \;\forall \;\; i=1,\ldots,Q, \;\; K=1,\ldots,K, \;\; j=1,\ldots,N, \;\; t=1,\ldots,T. \tag{6.8}$$

6.3.3 Homogeneous Student Model (HSM)

As explained, a serious drawback of most ITSs is the lack of ability to explain how learning occurs during students' interaction with such systems. Tensor-based models potentially can capture such dynamics in the interactions between students and questions overtime; however, one definition of learning is accumulation of knowledge overtime.Student models should be able to consider the fact that students improve their levels of knowledge and skills due to learning overtime. Indeed, a learning curve of each knowledge component illustrates the learning process and knowledge accumulation, therefore, one expects that the learning curve behaves as a non-decreasing function overtime. As mentioned above, interpretation of $\mathbf{V}$ is not straightforward, however, $\mathbf{C}^T\mathbf{V}$ shows student learning overtime. Hence, it is expected that learning curve exhibits an increasing or at least non-decreasing trend. As a matter of fact, there exist situations that students forget some of the course materials but the level of knowledge will not drastically change. In the matrix $\mathbf{C}^T\mathbf{V}$, rows represent students learning overtime so elements of the rows are expected to increase or at least show an increasing trend.

$$\sum_{k=1}^{K} C_{kj}V_{kt+1} - \sum_{k=1}^{K} C_{kj}V_{kt} \geq -\epsilon \quad \forall \;\; j=1\ldots,N, \quad t=1\ldots,T,$$

where ϵ is a small positive value that controls the forgetting rate. To estimate $\mathbf{W}$, $\mathbf{C}$, $\mathbf{V}$, $\boldsymbol{\theta}$ and $\mathbf{d}$, the following Maximum Likelihood Estimation (MLE) is employed for all observed data Ω_{obs}.

$$(P2): \quad \max_{\mathbf{W,C,V,d},\boldsymbol{\theta}} \quad \sum_{(i,j,t)\in\Omega_{obs}} log(P(\mathcal{Y}_{ijt}|\mathbf{w_i},\mathbf{c_j},\mathbf{v_t},d_i,\theta_j))$$

$s.t.$

$$\|\mathbf{w}_i\|_1 \leq \delta \ \ \forall \ \ i=1,\ldots,Q \tag{6.9}$$

$$\|\mathbf{w}_i\|_2 \leq \beta \ \ \forall \ \ i=1,\ldots,Q \tag{6.10}$$

$$\|\mathbf{C}\|_F = \xi \tag{6.11}$$

$$\|\mathbf{V}\|_F = \eta \tag{6.12}$$

$$\sum_{k=1}^{K} C_{kj}V_{kt+1} - \sum_{k=1}^{K} C_{kj}V_{kt} \geq -\epsilon \quad \forall \ \ j=1\ldots,N, \quad t=1\ldots,T, \tag{6.13}$$

$$W_{ki}, C_{kj}, V_{kt} \geq 0 \ \forall \ \ i=1,\ldots,Q, \ \ K=1,\ldots,K, \ \ j=1,\ldots,N, \ \ t=1,\ldots,T. \tag{6.14}$$

Constraints (6.9) to (6.12) guarantee that matrices **W,C,V** are sparse and the factorization problem has identifiability property. Set of constraints (6.13) ensure that students learn something and the level of knowledge does not deteriorate (it is improved or stays at the same level). Constraints (6.14) are non-negativity constraints.

6.3.4 Personalized Student Model (PSM)

This research offers a novel student model based on tensor factorization that considers student conceptual learning and personalizes dynamics of learning. The core-stone idea of PSM is that the data observed at different times should be treated differently. For example, a wrong answer at the beginning of the course is less informative than later in the course. Though HSM incorporates the knowledge acquisition in estimating parameters, it does not recognize that students learn with different rates. In addition, HSM assumes that students knowledge states (in whichever latent dimensions) tend to improve over time, one can also assume that if a student solves a problem correctly at one point in time, then she will also be able to solve it correctly at all the later time points (this assumption is a bit restrictive because in real educational systems, a student solves a problem correctly but he may not able to solve it few weeks after). Let $\mathcal{X}$ of size $Q \times N \times T$ denote the status of each student in each knowledge component overtime. One can factorize $\mathcal{Y}$ to matrix $\mathbf{W}$ and tensor $\mathcal{X}$ such that $\mathcal{Y}$ can be reconstructed by multiplying $\mathbf{W}$ and $\mathcal{X}$. Figure 6.2 shows the factorization of the observation matrix. As explained, $\mathbf{W}$ captures the relations between questions and knowledge components and $\mathcal{X}$ relates students, knowledge components and time. Similar to model I:

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j \quad \forall \quad i=1,\ldots,Q \ \ j=1,\ldots,N, \ t=1\ldots T \tag{6.15}$$

$$\mathcal{Y}_{ijt} \sim Ber(\Phi(\mathcal{T}_{ijt})),$$

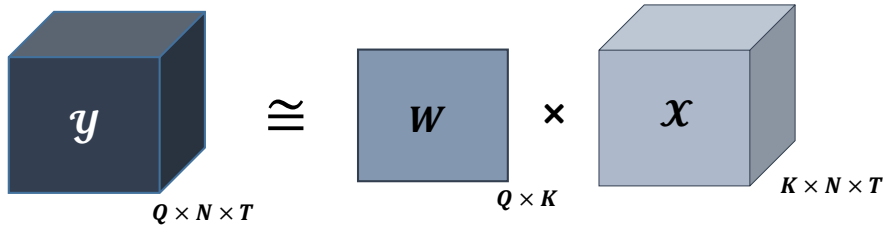

Figure 6.2: Tensor Factorization Model based on multiplication of $\mathbf{W}$ and $\mathbf{X}$.

where d_i, and θ_j denote the difficulty of question i and smartness of student j. *Ber(p)* shows a Bernoulli distribution with success probability p and $\Phi(x)$ is a logit function: Similar to SSM, the probability of $\mathcal{Y}_{ijt}$ given the parameters is defined as:

$$P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}} \tag{6.16}$$

The log likelihood estimation with constraints that are required to have a well-posed problem, is defined below:

$$(P3): \max_{\mathbf{W},\mathcal{X},v,\boldsymbol{\theta}} \sum_{(i,j,t)\in\Omega_{obs}} log(P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j))$$

$$s.t.$$

$$\|\mathbf{w}_i\|_1 \leq \delta \ \ \forall \ \ i = 1, \ldots, Q \tag{6.17}$$

$$\|\mathbf{w}_i\|_2 \leq \beta \ \ \forall \ \ i = 1, \ldots, Q \tag{6.18}$$

$$\|\mathcal{X}_{::t}\|_F = \xi \ \ \forall \ \ t = 1, \ldots, T \tag{6.19}$$

$$\mathcal{X}_{kjt} \leq \mathcal{X}_{kjt+1} \ \forall \ \ k = 1, \ldots, K, \ \ j = 1, \ldots, N, \ t = 0, \ldots, T-1 \tag{6.20}$$

$$W_{ik} \geq 0 \ \forall \ \ \ i = 1, \ldots, Q, \ \ K = 1, \ldots, K, \tag{6.21}$$

$$\mathcal{X}_{kj0} \geq 0 \ \ \forall \ \ k = 1, \ldots, K, \ \ j = 1, \ldots, N. \tag{6.22}$$

In PSM, constraints (6.17) and (6.18), similar to those of SSM and HSM, ensure the sparsity of matrix $\mathbf{W}$. Constraints (6.19) prevent $\mathcal{X}$ from unbounded growth. Constraints (7.10) play an important role in increasing expressive power of this model and guarantee knowledge accumulation. One can interpret $\mathcal{X}$ as learning curve of students in knowledge components. The elements of $\mathcal{X}$ are not necessarily similar which assume personalized learning curve for each student.

One incisive critique of tensor factorization is the expensive operations due to extra dimensions compared to matrix factorization, However, the above-mentioned low rank property, $K << Q, N$, makes $\mathcal{X}$ a narrow tensor, therefore, no extra computational costs are imposed to the model.

6.4 Scalable Algorithms for Student Models

Formulating the problem of personalizing learning using tensor factorization may bring new levels of flexibility and/or quality into ITSs, however there are challenges that need to be

considered. Similar to matrix factorization, the traditional tensor factorization can be formulated as a convex optimization problem, however, extensions of tensor factorization make the optimization task significantly more difficult Liu et al. (2013). The success of models introduced in the previous sections in practice, specially in large-scale ITSs, i.e., MOOCs, highly depends upon implementing efficient optimization algorithms to find a good enough solution. The optimization problems described in (P1), (P2) and (P3) are multi-convex and non-smooth which need powerful and scalable algorithm to attack MLE. Gorski et al. (2007).

6.4.1 SSM Optimization Problem

Distinguished from convex optimization problems, multi-convex problems are in general global optimization problems which may contain a large number of local minima Gorski et al. (2007). The constraints of (P1) are a convex set and one can use Lagrangian multipliers to convert the problem into an unconstrained multi-convex optimization. Non-negativeity constraints are trivial to satisfy, therefore, the optimization model is as follows:

$$
\begin{aligned}
(P_1')\ \min_{\mathbf{W,C,V,d},\boldsymbol{\theta}} \Gamma = & \sum_{(i,j,t)\in\Omega_{obs}} -log(P\mathcal{Y}_{ijt}|\mathbf{w_i},\mathbf{c_j},\mathbf{v_t},d_i,\theta_j)) + \lambda_1 \sum_{i=1}^{Q} \|\mathbf{w}_i\|_1 \\
& + \frac{\lambda_2}{2} \sum_{i=1}^{Q} \|\mathbf{w}_i\|_2 + \frac{\lambda_3}{2}\|\mathbf{C}\|_F + \frac{\lambda_4}{2}\|\mathbf{V}\|_F
\end{aligned}
\tag{6.23}
$$

$st:$
$W_{ki}, C_{kj}, V_{kt} \geq 0 \ \forall \ \ i = 1,\ldots,Q, \ \ K = 1,\ldots,K, \ \ j = 1,\ldots,N, \ \ t = 1,\ldots,T.$

To minimize (P_1'), gradient-based methods are used so the gradients of P_1 are calculated below:

$$\nabla\Gamma_{W_{ki}} = -\sum_{j=1}^{N}\sum_{t=1}^{T} C_{kj}V_{kt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \frac{\lambda_2 W_{ki}}{\|W\|_F} \tag{6.24}$$

$$\nabla\Gamma_{C_{kj}} = -\sum_{i=1}^{Q}\sum_{t=1}^{T} W_{ki}V_{kt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \frac{\lambda_3 C_{kj}}{\|C\|_F} \tag{6.25}$$

$$\nabla\Gamma_{V_{kt}} = -\sum_{j=1}^{N}\sum_{i=1}^{Q} C_{kj}W_{ki}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \frac{\lambda_4 V_{ki}}{\|V\|_F} \tag{6.26}$$

$$\nabla\Gamma_{d_i} = \sum_{j=1}^{N}\sum_{t=1}^{T} (\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) \tag{6.27}$$

$$\nabla\Gamma_{\theta_j} = -\sum_{i=1}^{Q}\sum_{t=1}^{T} (\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) \tag{6.28}$$

6.4.2 HSM Optimization Problem

Similar to SMM, all gradients of HSM are obtained using (6.24) to (6.28). However, constraints (6.13) need to be treated differently.

$$(P_2')\ \min_{\mathbf{W,C,V,d},\boldsymbol{\theta}} F = \sum_{(i,j,t)\in\Omega_{obs}} -log(P(\mathcal{Y}_{ijt}|\mathbf{w_i},\mathbf{c_j},\mathbf{v_t},d_i,\theta_j)) + \lambda_1 \sum_{i=1}^{Q} \|\mathbf{w}_i\|_1 + \frac{\lambda_2}{2}\sum_{i=1}^{Q}\|\mathbf{w}_i\|_2 + \frac{\lambda_3}{2}\|\mathbf{C}\|_F + \frac{\lambda_4}{2}\|\mathbf{V}\|_F \tag{6.29}$$

$$st:$$
$$\sum_{k=1}^{K} C_{kj}V_{kt+1} - \sum_{k=1}^{K} C_{kj}V_{kt} \geq -\epsilon \quad \forall\ j = 1\ldots,N, \quad t = 1\ldots,T,$$
$$W_{ki}, C_{kj}, V_{kt} \geq 0\ \forall\ i = 1,\ldots,Q,\ \ K = 1,\ldots,K,\ \ j = 1,\ldots,N,\ \ t = 1,\ldots,T.$$

6.4.3 PSM Optimization Problem

PSM offers a new tensor factorization model, however, the structure of optimization problem remains almost the same as (P_1') and (P_2'). Constraints (7.10) and non-negativity constraints are straightforward to handle.

$$(P_3')\ \min_{\mathbf{W,C,V}} \Gamma = \sum_{(i,j,t)\in\Omega_{obs}} -log(P(\mathcal{Y}_{ijt}|\mathbf{w_i},\mathcal{X}_{:jt})) + \lambda_1\sum_{i=1}^{Q}\|\mathbf{w}_i\|_1 + \frac{\lambda_2}{2}\sum_{i=1}^{Q}\|\mathbf{w}_i\|_2 + \frac{\lambda_3}{2}\sum_{t=1}^{T}\|\mathcal{X}_{::t}\|_F$$

$$st:$$
$$\mathcal{X}_{kjt} \leq \mathcal{X}_{kjt+1}\ \forall\ \ k = 1,\ldots,K,\ \ j = 1,\ldots,N,\ t = 0,\ldots,T-1 \tag{6.30}$$
$$W_{ik} \geq 0\ \forall\ \ \ i = 1,\ldots,Q,\ \ K = 1,\ldots,K, \tag{6.31}$$
$$\mathcal{X}_{kj0} \geq 0\ \ \forall\ \ k = 1,\ldots,K,\ \ j = 1,\ldots,N \tag{6.32}$$

Derivatives of (P_3') are similar to those of (P_1'), however, since the factorization is different, one can utilize (6.33) and (6.34) to calculate $\nabla\Gamma_W$:

$$\nabla\Gamma_{W_{ki}} = -\sum_{j=1}^{N}\sum_{t=1}^{T}\mathcal{X}_{kjt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \lambda_2\frac{W_{ki}}{\|W\|_F} \tag{6.33}$$

$$\nabla\Gamma_{\mathcal{X}_{kjt}} = -\sum_{i=1}^{Q} W_{ki}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \lambda_3\sum_{t=1}^{T}\frac{\mathcal{X}_{kjt}}{\|\mathcal{X}_{::t}\|_F} \tag{6.34}$$

One can use (6.27) and (6.28) to calculate $\nabla\Gamma_{\mathbf{d}}$ and $\nabla\Gamma_{\boldsymbol{\theta}}$, respectively considering the fact that $\mathcal{T}$ is computed using (7.5).

6.4.4 Block Coordinate Descent

Block Coordinate Descent (BCD) methodsWright (2015) are extensions of Coordinate Descent (CD) algorithms that have been developed to tackle optimization problems. CD is a well-studied optimization technique which has been successfully employed to tackle with a variety of large-scale machine learning problems such as Support Vector Machines Hsieh et al. (2008), maximum entropy models Hsieh et al. (2008), Non-negative Matrix Factorization problems Cichocki and Anh-Huy (2009) and so on. The main idea of CD is to update a single variable at each iteration while keeping others unchanged. However, most applications utilize BCD methods which update groups of blocks of variables at each iteration, thus searching occurs along a coordinate hyperplane rather than a single coordinate directionWright (2015). Most concepts of single-coordinate descent methods can be generalized to the BCD without any fundamental changes Wright (2015).

BCD has been reported as an efficient, scalable algorithm for large-scale convex optimization problems while optimization problems proposed in this paper are non-convex and non-smooth due to ℓ-1 regularization. However, one can adopt BCD to (P_1') and use projection methods to handle non-smoothness and satisfy non-negativity constraints. If non-negativity constraints are satisfied, the absolute value of elements in ℓ-1 can be dropped and have a differential function. In addition, ℓ-2 regularization is computationally expensive and differentiation is difficult so can be relaxed to Frobenius norm since $\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$.

The original BCD algorithm cyclically updates each block of variables by exactly solving the sub-problem, however, the blocks can be updated in different orders (e.g. essentially cyclic, randomized or parallel) and solve the sub-problem inexactly. In this implementation, indexes $i \in \{1, \ldots, Q\}$, $j \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$ are randomly selected and a block of variables grouped by k are updated in each iteration as follows:

$$\mathbf{w}_i^{new} \leftarrow \max\{\mathbf{0}, \mathbf{w}_i^{old} - \beta \nabla \Gamma_{\mathbf{w}_i}\},$$

where $\mathbf{w}_i$ is a vector of variables related to question i. To satisfy non-negativity constraints, a projection operation is used. To update $\mathbf{c}_j$, the rule below is performed.

$$\mathbf{c}_j^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{c}_j^{old} - \beta \nabla \Gamma_{\mathbf{c}_j}\}\Big),$$

with block variable $\mathbf{c}_j$. Projection is used to satisfy non-negativity constraints and also $\mathbf{c}_j$ is scaled to prevent it form unbounded growth. Similarly, $\mathbf{v}_t$ is updated as follows:

$$\mathbf{v}_t^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{v}_t^{old} - \beta \nabla \Gamma_{\mathbf{v}_t}\}\Big).$$

Updating d_i and θ_j is trivial. Algorithm (5) for SSM describes the BCD steps.

6.4.5 Alternating Direction Methods of Multipliers

Alternating Direction Methods of Multipliers (ADMM) belongs to the class of BCD methods. ADMM is an algorithm that is designed to combine the decomposability of dual ascent with the fast convergence properties of the method of multipliers Boyd et al. (2011). In this

Algorithm 5 Block Coordinate Descent Algorithm-SSM

Input: Observed Tensor $\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$.
Output: Completed Tensor $\mathcal{Y}$, Matrices $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
BlockCoordinateDescent()

1. Initialize randomly $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
2. **while** (stopping criteria) **do**
3. randomly select $i \in \{1, \ldots, Q\}$, $j \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$;
4. calculate $\nabla\Gamma_{\mathbf{w}_i}$, $\nabla\Gamma_{\mathbf{c}_j}$, $\nabla\Gamma_{\mathbf{v}_t}$, $\nabla\Gamma_{d_i}$ and $\nabla\Gamma_{\theta_j}$ using (6.24)-(6.28)
5. update $\mathbf{w}_i^{new} \leftarrow \max\{\mathbf{0}, \mathbf{w}_i^{old} - \beta\nabla\Gamma_{\mathbf{w}_i}\}$
6. update $\mathbf{c}_j^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{c}_j^{old} - \beta\nabla\Gamma_{\mathbf{c}_j}\}\Big)$
7. update $\mathbf{v}_t^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{v}_t^{old} - \beta\nabla\Gamma_{\mathbf{v}_t}\}\Big)$
8. update $d_i^{new} \leftarrow d_i^{old} - \beta\nabla\Gamma_{d_i}$
9. update $\theta_j^{new} \leftarrow \theta_j^{old} - \beta\nabla\Gamma_{\theta_j}$
10. calculate objective function using (6.23)
11. **end**
12. report AIC, AIC_c, BIC, MSE

algorithm, each category of variables (i.e., $\mathbf{W}, \mathbf{C}$ and $\mathbf{V}$ corresponding to a sub-problem are updated in an alternating fashion. The ideal situation is that the optimal solution for each sub-problem is found. ADMM implemented for SSM is explained in Algorithm (6). As explained, $(P_2{}')$ and $(P_3{}')$ have constraints. In case of $(P_2{}')$, the easiest way is to replace (6.13) with $\mathbf{v}_t \leq \mathbf{v}_{t+1}$. In $(P_3{}')$, one can use the following projection to satisfy (7.10).

$$\mathcal{X}^{new}_{::t+1} = \max\left\{\mathcal{X}_{::t}, \mathcal{X}^{new}_{::t+1}\right\}$$

Algorithm 6 Alternating Direction Method of Multipliers-SSM

Input: Observed Tensor $\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$.
Output: Completed Tensor $\mathcal{Y}$, Matrices $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
AlternatingDirectionMethodMultipliers()

1. Initialize randomly $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
2. **while** (stopping criteria) **do**
3. choose step sizes β_W, β_d, β_C, β_θ and β_V
4. **while** (stopping criteria) **do**
5. calculate $\nabla\Gamma_{\mathbf{w}}$ and $\Gamma_{\mathbf{d}}$ using (6.24) and (6.27)
6. update $\mathbf{w}^{new} \leftarrow \max\{\mathbf{0}, \mathbf{w}^{old} - \beta_W \nabla\Gamma_{\mathbf{w}}\}$
7. update $\mathbf{d}^{new} \leftarrow \mathbf{d}^{old} - \beta_d \nabla\Gamma_{\mathbf{d}}$
8. **while** (stopping criteria) **do**
9. calculate calculate $\nabla\Gamma_{\mathbf{c}}$ and $\nabla\Gamma_{\boldsymbol{\theta}}$ using (6.25) and (6.28)
10. update $\mathbf{C}^{new} \leftarrow \frac{1}{1+\beta_C\gamma}\Big(\max\{\mathbf{0}, \mathbf{C}^{old} - \beta_C \nabla\Gamma_{\mathbf{C}}\}\Big)$
11. update $\boldsymbol{\theta}^{new} \leftarrow \boldsymbol{\theta}^{old} - \beta_\theta \nabla\Gamma_{\boldsymbol{\theta}}$
12. **while** (stopping criteria) **do**
13. calculate $\nabla\Gamma_{\mathbf{V}}$ using (6.26) and (6.28)
14. update $\mathbf{V}^{new} \leftarrow \frac{1}{1+\beta_V\gamma}\Big(\max\{\mathbf{0}, \mathbf{V}_t^{old} - \beta \nabla\Gamma_{\mathbf{V}}\}\Big)$
15. calculate objective function using (6.23)
16. **end**
17. report AIC, AIC_c, BIC, MSE

6.5 Experimental Analysis

This section describes the process of generating and preparing synthetic data as well as experiments setup and implementation of algorithm. There are several datasets that researchers in this area use to test their algorithms, however, one needs grand truth data representing the state of knowledge of students to be able to measure the algorithm performance and interpretability of the models.

In order to test the proposed models, both synthetic data (as described in Chapter 4) with different sizes are employed. The problems are categorized into 3 classes: 1) Small-size problems 2)Medium-size problems 3)Large- size problems. SSM and HSM have $(N + Q + T)K+(N+Q)$ variables in the optimization model while PSM includes $K(NT+Q)+(N+Q)$ variables in the optimization model.

Table 6.1: Experiments setup for testing Tensor Factorization Models.

Category	Experiment ID	Question	Student	Time (week)	KC	Sparsity	Estimated KC	Iterations	Num of Run
Small	1	100	20	6	3	25%	2-5	500	30
	2	100	20	6	3	50%	2-5	500	30
	3	100	20	6	3	75%	2-5	500	30
	4	100	20	6	3	85%	2-5	500	30
Medium	5	200	40	10	5	15%	2-5	200	25
	6	200	40	10	5	25%	2-5	200	25
	7	3200	40	10	5	50%	2-5	200	25
	8	200	40	10	5	75%	2-5	200	25
Large	9	500	100	15	8	10%	2-5	100	20
	10	500	100	15	8	15%	2-5	100	20
	11	500	100	15	8	25%	2-5	100	20
	12	500	100	15	8	50%	2-5	100	20

6.5.1 Implementation

To implement the BCD and ADMM algorithms, the following aspects were used:

- **Software:** We used a custom code written Python using NumPy, CVXPY and Pandas. Double precision values were used for matrices and tensors. The observation matrix $\mathcal{Y}$ is binary and highly sparse so the compressed column and row representations is recommended.

- **Hardware:** Different hardware for various experimental setup was used. For experiments 4 to 8, we used a desktop with Intel(R) Core(TM)i5 3GHz processor, 8GB RAM and 64 bit operating system. For experiments 9 to 12, Dell machines with 12x2.40GHz Intel Xeon E5645 Processor Cores and 12 cores per node with 48 GB memory and Linux (RedHat Enterprise Linux 6.1 2.6.32 Kernel) operating systems were used.

- **Reporting:** The main challenge in dealing with high dimension non-convex optimization problems is the huge number of local optima causing the majority iteration-based algorithms get stuck and not converge to the global optimum solution. Anther issue with such large-scale optimization problems is that the optimal solution is usually unknown particularly in real-world applications. Consequently, defining a stopping

criterion is problematic even though one knows the optimal solution. In this paper, we consider number of iterations as the stopping criteria. With the number of iterations, we fix the number of iterations and ADMM reports the best found solution. To ensure that random initial conditions do not affect the results, each experiment is run several times based on Table 6.1. In addition, the time to find the best solution from initialization is captured and reported. In order to provide better insights about the models and help one to compare the models, criteria such as Akaike Information Criteria (AIC), AIC corrected (AIC_c), Bayesian Information Criteria (BIC), Log likelihood, Frobenius Norm and Mean Square Error (MSE) described below are reported.

$$AIC = 2k - 2\Gamma, \quad AIC_c = AIC + \frac{2k(k+1)}{n-k-1}, \quad BIC = -2\Gamma + k\log(n),$$

where k is the number of parameters, Γ denotes the loglikelihood function and n indicates the sample size. The next section explores the performance of BCD and ADMM on small-, medium- and large-sized problems of (P1), P(2) and P(3). MSE which is considered as the main criterion to evaluate the performance of model, compares the probabilities of answering questions obtained from synthetic data process against the estimated probabilities for the test data (removed randomly from the synthetic data based on the sparsity rate).

6.5.2 Comparison between BCD and ADMM

As discussed before, though both BCD and ADMM belong to the same family of iterative gradient-based algorithms, there are some differences which make them appropriate for different applications. In many real-time applications, the run-time is very important and the implemented methods should provide a good-enough solution in short amount of time. Figure 6.3 compares the convergence rate of ADMM against BCD over a small-sized problem with different knowledge components. Because ADMM solves sub-optimal problems in each step, it has a faster convergence rate in terms of number of iterations. On the other side, each iteration takes more time. In this study, the quality of solutions is crucial to evaluate performance of proposed models, therefore, ADMM is selected for all instances.

6.5.3 Simulation Results

This section reports the simulation results and compares the proposed models with different measures. To choose the optimization algorithms, ADMM is compared against BDC in small-sized problem with the same initial condition (see Figure 6.3. Due to the performance of ADMM, all three models are trained using ADMM. As discussed before, there is a trade-off between time and performance of ADMM and BCD that one needs to take into account particularly for real-time applications.

Results of Small-sized Instances

Small-sized instances include problems with 20 students and 100 questions that represent a regular course in STEM, specially in winter or summer semesters. Table 6.2 summarizes the results of simulations of predicting students performance on small-sized instances and

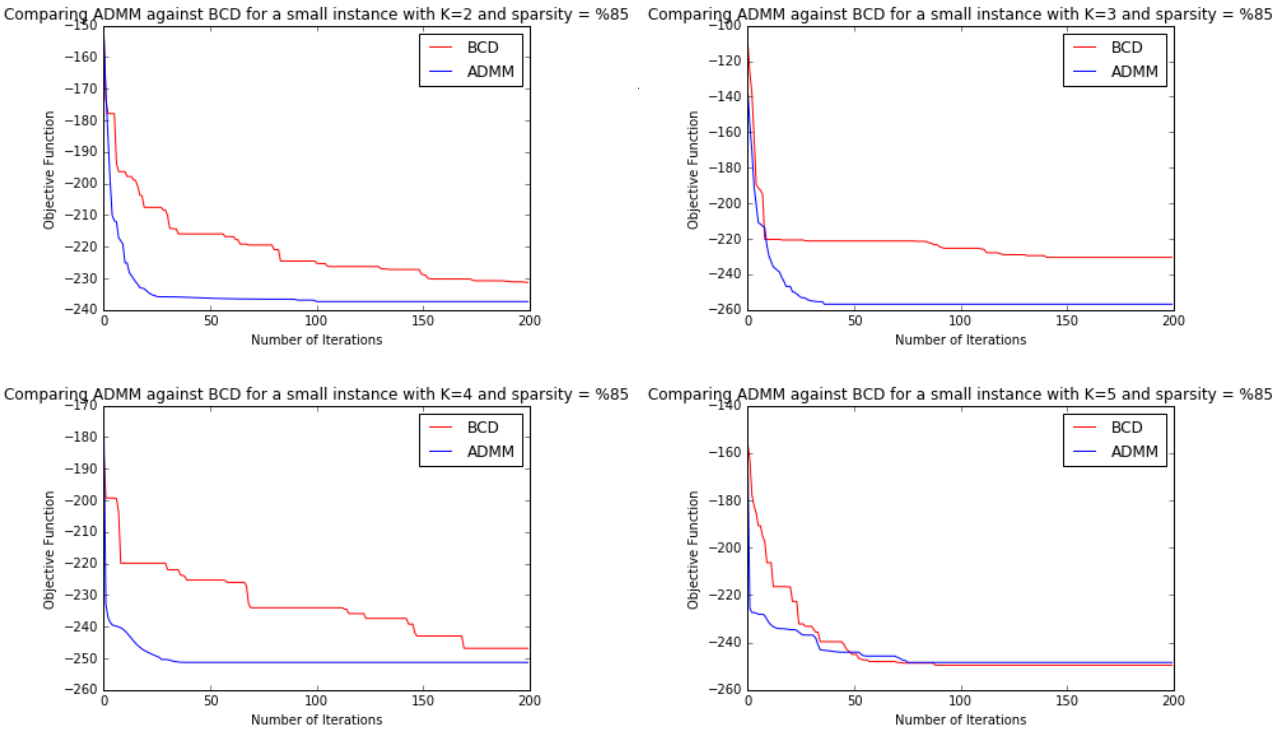

Figure 6.3: Convergence rate of both BCD and ADMM algorithms on small-sized with different knowledge components in 200 iterations which shows ADMM converges with less number of iterations.

reports different indexes to compare the models. The results show that PSM outperforms SSM and HSM and provides better predictions. To more closely study the performance of the models, the performance of models based on MSE measure for test set is compared. Figure 6.4 depicts MSE of test set in small-sized instances with different knowledge components and 25% sparsity. PSM exhibits a robust performance because of the small variance of MSE captured in the box-plot. In small-sized instances with $K = 2$ and $K = 3$, the performance of HSM and PSM are close however, the averages of MSE are statistically different. Similarly, the same comparison is reported with considering 85% sparsity in the observation (see Figure 6.5). PSM also excels SSM and HSM in this rate of sparsity and exhibits a robust performance. Figure 6.6 illustrates the performance of PSM with regard to sparsity rates. As the sparsity increases, MSE increases as well.

Results of Medium-sized Instances

Medium-sized instances contain 200 questions and 40 students, a medium size STEM class in Fall and Spring semesters. Figure 6.7 depicts MSE of the test set obtained from the experiment with sparsity rate 75% and different number of knowledge components. PSM shows superiority performance compared to SSM and HSM. The MSE variance of PSM is very small. Performance of HSM is also acceptable, however, the variance is high. Table 6.3 summarizes results of running all three models over medium-sized instances. The time average to obtain a good-enough solution is reasonable and the models can quickly update the parameters after new student activities are observed.

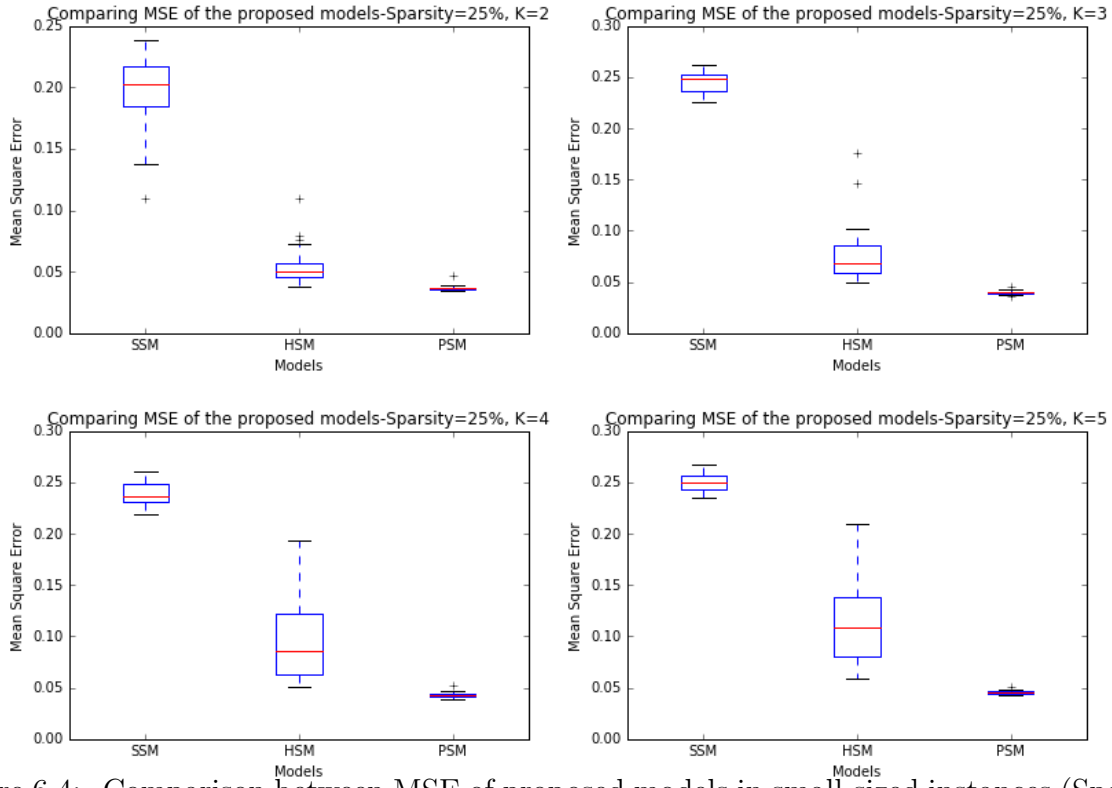

Figure 6.4: Comparison between MSE of proposed models in small-sized instances (Sparsity rate is 25%) .

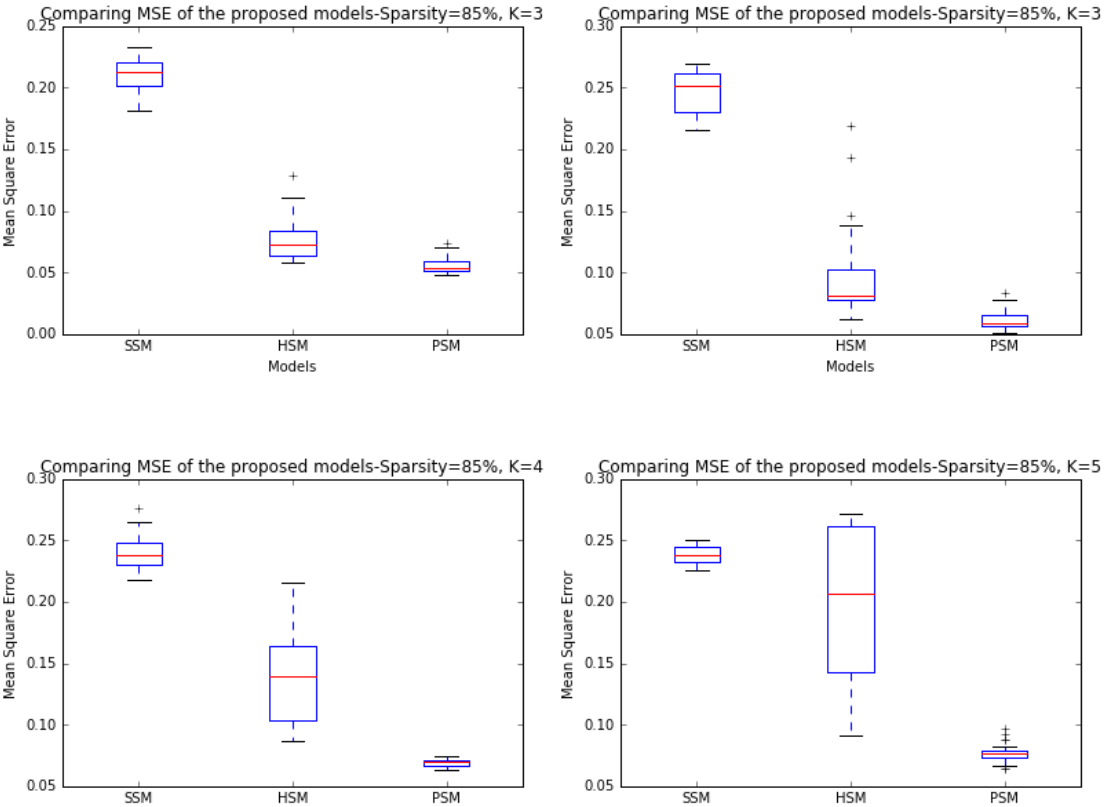

Figure 6.5: Comparison between MSE of proposed models in small-sized problem (Sparsity rate is 85%) .

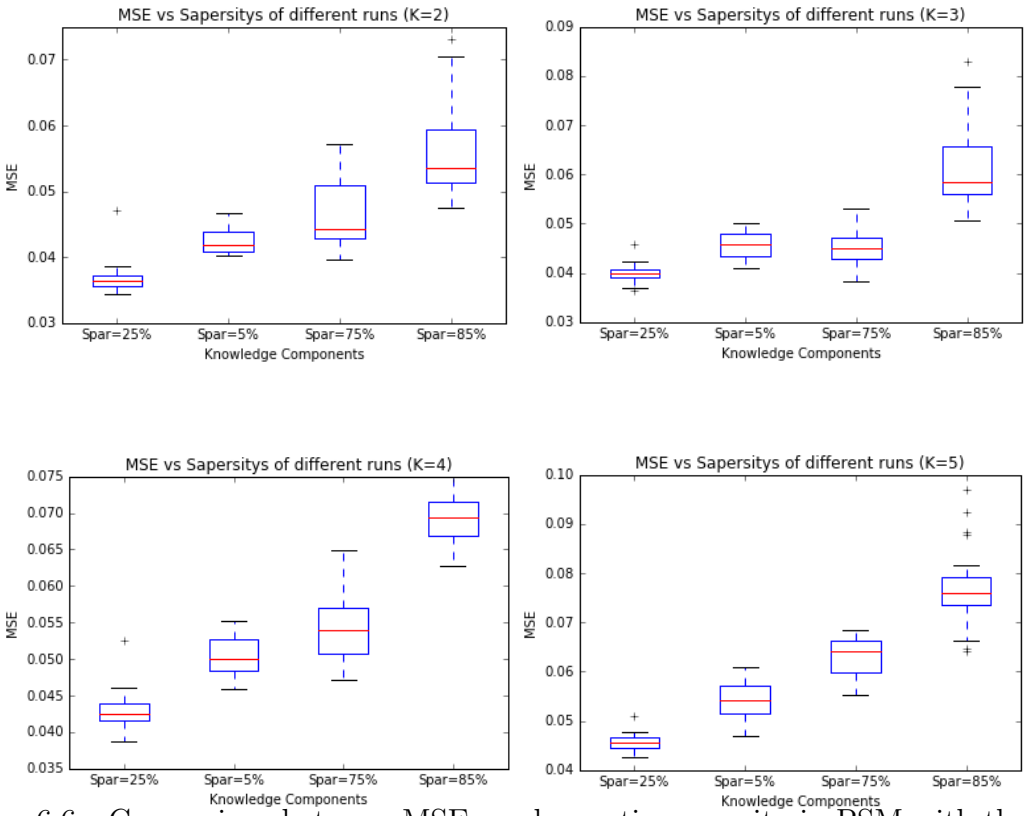

Figure 6.6: Comparison between MSE vs observation sparsity in PSM with the small-sized problem (different number of Knowledge Component) .

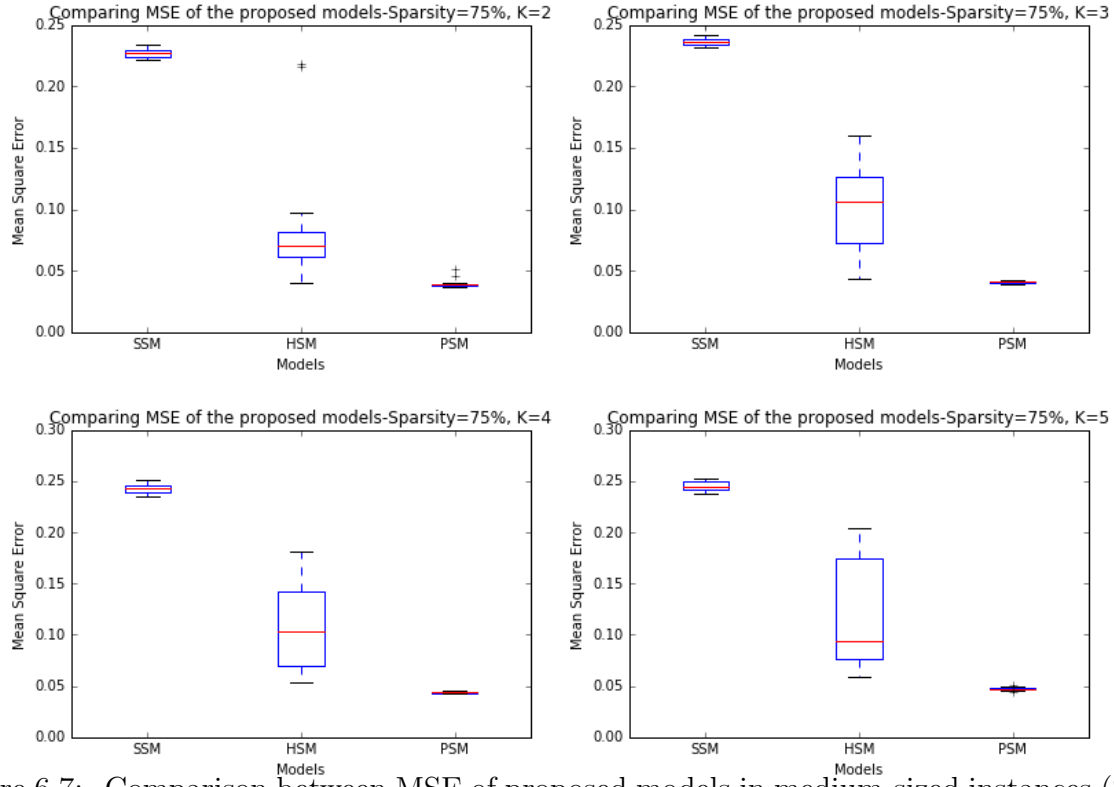

Figure 6.7: Comparison between MSE of proposed models in medium-sized instances (Sparsity rate is 25%) .

Table 6.2: Simulation Results of the small size problem using three models, different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE	
SSM	25%	2	8679.249	8690.104	10542.2	145.895	-4087.625	48.268	0.195	
		3	8718.492	8743.147	11512.918	75.858	-3981.246	52.83	0.233	
		4	9660.322	9799.774	16165.865	47.271	-3950.161	53.615	0.24	
		5	10147.78	10370.02	18279.708	69.988	-3973.89	54.718	0.25	
	50%	2	5283.648	5317.222	8536.419	17.991	-2201.824	50.704	0.214	
		3	5660.451	5737.399	10539.608	31.581	-2170.225	54.208	0.245	
		4	6179.3	6318.751	12684.842	43.009	-2209.65	53.581	0.24	
		5	6632.397	6854.638	14764.325	39.694	-2216.198	54.157	0.245	
	75%	2	3110.927	3144.501	6363.698	2.274	-1115.463	50.989	0.217	
		3	3520.042	3596.99	8399.199	28.566	-1100.021	54.272	0.246	
		4	3923.637	4063.088	10429.179	18.181	-1081.818	53.95	0.243	
		5	4411.602	4633.842	12543.53	33.268	-1105.801	54.722	0.25	
	85%	2	2303.694	2337.268	5556.465	2.82	-711.847	50.269	0.211	
		3	2641.75	2718.699	7520.907	21.594	-660.875	54.393	0.247	
		4	3101.327	3240.778	9606.87	17.535	-670.664	53.736	0.241	
		5	3530.748	3752.988	11662.676	4.944	-665.374	53.468	0.238	
HSM	25%	2	9276.795	9287.65	11139.746	3.015	-4386.398	25.284	0.054	
		3	9412.838	9437.493	12207.264	4.767	-4328.419	30.284	0.079	
		4	9592.628	9636.911	13318.529	5.776	-4292.314	33.02	0.094	
		5	9753.432	9823.365	14410.81	5.626	-4246.716	35.942	0.111	
	50%	2	5349.594	5360.449	7212.545	5.585	-2422.797	29.762	0.075	
		3	5438.839	5463.494	8233.265	2.761	-2341.419	34.475	0.101	
		4	5904.906	5949.189	9630.807	50.791	-2448.453	24.579	0.05	
		5	5901.79	5971.722	10559.167	8.648	-2320.895	40.015	0.136	
	75%	2	2928.458	2939.313	4791.409	15.013	-1212.229	35.766	0.109	
		3	3120.603	3145.259	5915.029	10.434	-1182.302	35.608	0.109	
		4	3308.058	3352.341	7033.959	4.797	-1150.029	43.453	0.16	
		5	3548.754	3618.687	8206.131	5.608	-1144.377	47.822	0.192	
	85%	2	2067.214	2078.069	3930.165	6.323	-781.607	30.376	0.078	
		3	2204.303	2228.959	4998.73	7.179	-724.152	34.175	0.101	
		4	2454.114	2498.397	6180.015	4.366	-723.057	40.693	0.14	
		5	2698.58	2768.512	7355.957	14.807	-719.29	48.016	0.198	
PSM	25%	2	9356.938	9367.793	11219.889	86.625	-4426.469	20.999	0.037	
		3	9610.989	9635.644	12405.415	67.407	-4427.494	21.912	0.04	
		4	9862.182	9906.466	13588.084	106.764	-4427.091	22.709	0.043	
		5	10130.819	10200.751	14788.196	165.745	-4435.41	23.435	0.046	0.05
	50%	2	5423.435	5434.29	7286.386	30.186	-2459.718	22.584	0.043	
		3	5617.815	5642.471	8412.241	56.479	-2430.907	23.383	0.046	
		4	5904.906	5949.189	9630.807	50.791	-2448.453	24.579	0.05	
		5	6152.351	6222.283	10809.728	60.314	-2446.175	25.552	0.054	
	75%	2	2992.996	3003.851	4855.947	17.98	-1244.498	23.622	0.047	
		3	3212.147	3236.803	6006.573	19.052	-1228.073	23.201	0.045	
		4	3479.487	3523.771	7205.389	23.221	-1235.744	25.516	0.054	
		5	3738.483	3808.416	8395.86	27.574	-1239.242	27.551	0.063	
	85%	2	2081.838	2092.693	3944.789	16.051	-788.919	25.87	0.056	
		3	2239.775	2264.431	5034.201	24.134	-741.887	27.074	0.061	
		4	2517.483	2561.766	6243.384	23.929	-754.741	28.813	0.069	
		5	2777.209	2847.141	7434.586	31.47	-758.605	30.419	0.077	

Results of Large-sized Instances

Large-sized instances encompass 500 questions and 100 students. Table 6.4 summarizes the results of simulations for large-sized instances. PSM outperforms in majority of experiments

Table 6.3: Simulation Results of the Medium-sized problem using three models, different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE
SSM	15%	2	51720.36	51726.662	56365.251	41.861	-25360.18	123.495	0.191
		3	51481.741	51495.955	58449.077	49.633	-24990.87	129.307	0.211
		5	51876.305	51916.02	63488.532	65.098	-24688.152	129.248	0.209
	25%	2	46338.534	46344.836	50983.425	59.443	-22669.267	115.235	0.167
		3	47008.48	47022.695	53975.817	72.995	-22754.24	114.693	0.165
	50%	2	32200.593	32206.895	36845.484	9.759	-15600.29	128.825	0.209
		3	34394.879	34477.79	51116.486	840.592	-15397.439	108.719	0.148
		4	34944.438	35092.956	57239.915	1156.122	-15072.219	122.462	0.188
		5	35223.168	35457.015	63092.514	1122.136	-14611.584	131.707	0.217
	75%	2	16843.459	16880.038	27991.197	10.188	-7221.729	134.812	0.227
		3	17838.036	17920.947	34559.643	8.001	-7119.018	137.612	0.237
		4	18934.271	19082.788	41229.747	9.534	-7067.135	139.363	0.243
		5	20241.866	20475.713	48111.212	23.348	-7120.933	140.022	0.245
HSM	15%	2	53686.504	53692.806	58331.395	13.148	-26343.251	111.741	0.158
		3	53179.076	53193.291	60146.412	17.737	-25839.537	126.765	0.203
		5	53693.702	53733.417	65305.929	34.406	-25596.851	122.182	0.187
	25%	2	46899.026	46905.328	51543.917	4.05	-22949.512	130.435	0.213
		3	47851.718	47865.933	54819.054	21.942	-23160.033	109.87	0.152
	50%	2	33805.685	33842.264	44953.423	624.673	-15702.84	94.0	0.110
		3	32672.503	32686.717	39639.839	18.991	-15586.251	101.231	0.129
		4	32838.284	32863.626	42128.066	25.508	-15419.14	114.632	0.168
		5	33088.197	33127.912	44700.425	37.338	-15294.099	111.113	0.157
	75%	2	17087.543	17093.845	21732.434	15.583	-8043.771	80.178	0.086
		3	17328.486	17342.701	24295.823	22.045	-7914.243	89.804	0.104
		4	17614.223	17639.565	26904.005	34.818	-7807.111	92.392	0.111
		5	18095.392	18135.107	29707.619	53.063	-7797.696	94.517	0.117
PSM	15%	2	58163.54	58200.12	69311.279	328.617	-27881.77	93.518	0.11
		3	546752.793	546849.584	615919.764	1007.521	-267376.397	394.482	0.208
		5	525277.366	525547.664	640555.651	280.498	-252638.683	430.52	0.247
	25%	2	50399.653	50436.232	61547.391	397.934	-23999.826	100.289	0.1267
		3	51811.578	51894.49	68533.185	704.491	-24105.789	101.814	0.13
	50%	2	32888.025	32894.327	37532.916	85.483	-15944.013	76.411	0.075
		3	33439.091	33453.306	40406.427	112.309	-15969.545	78.087	0.077
		4	33912.091	33937.433	43201.873	151.735	-15956.046	81.622	0.0864
		5	34254.91	34294.625	45867.138	146.107	-15877.455	81.856	0.087
	75%	2	17326.508	17332.81	21971.399	145.163	-8163.254	56.334	0.04
		3	17911.154	17925.368	24878.49	211.327	-8205.577	57.001	0.0417
		4	18232.683	18258.025	27522.465	314.742	-8116.342	59.05	0.044
		5	18727.232	18766.946	30339.459	420.525	-8113.616	61.324	0.047

with different sparsity rates and number of knowledge components. Only in one case, PSM performs as well as SSM and HSM.

6.6 Conclusion

This chapter presents three tensor factorization-based student models that improves the state-of-the-art in Latent Factor Analysis by offering personalized models that take into account the differences between students in learning different contents over time. The proposed

Table 6.4: Simulation Results of the large-sized problem using three models, different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE
SSM	10%	2	514175.774	514179.819	528355.004	229.606	-255857.887	407.214	0.221
		3	511094.188	511103.292	532363.031	310.869	-253702.094	415.166	0.23
		4	4494268.81	494285.008	522627.268	308.655	-244674.405	436.418	0.254
		5	496029.117	496054.444	531477.189	351.132	-244939.558	432.949	0.25
	15%	2	484116.441	484120.485	498295.67	216.091	-240828.22	410.84	0.225
		3	488403.692	488412.796	509672.535	283.959	-242356.846	406.34	0.22
		4	481374.514	481390.711	509732.972	420.323	-238227.257	417.623	0.233
		5	474968.952	474994.279	510417.025	363.872	-234409.476	425.338	0.241
	25%	2	429330.372	429334.416	443509.601	192.793	-213435.186	404.444	0.218
		3	432443.819	432452.924	453712.663	253.448	-214376.91	401.761	0.215
		4	3408442.98	408459.177	436801.438	575.587	-201761.49	441.391	0.26
		5	407819.662	407844.989	443267.735	276.233	-200834.831	444.862	0.264
	50%	2	276894.218	276898.262	291073.447	80.779	-137217.109	413.082	0.228
		3	278877.231	278886.335	300146.074	115.8	-137593.615	412.201	0.227
		4	273707.644	273723.842	302066.102	256.938	-134393.822	444.707	0.264
		5	277969.74	277995.067	313417.813	295.091	-135909.87	434.777	0.253
HSM	10%	2	485733.714	485737.758	499912.943	60.816	-241636.857	423.958	0.24
		3	493836.421	493845.526	515105.264	86.101	-245073.21	427.841	0.244
		4	490247.312	490263.51	518605.77	94.76	-242663.656	436.371	0.254
		5	499815.053	499840.38	535263.126	97.409	-246832.527	433.823	0.251
	15%	2	459147.227	459151.272	473326.456	57.034	-228343.614	420.423	0.236
		3	465612.334	465621.438	486881.177	53.634	-230961.167	428.786	0.245
		4	469046.48	469062.678	497404.939	77.333	-232063.24	431.188	0.248
		5	470529.395	470554.722	505977.468	107.274	-232189.698	434.033	0.251
	25%	2	407597.46	407601.504	421776.689	57.18	-202568.73	423.297	0.239
		3	410613.839	410622.944	431882.683	56.184	-203461.92	428.311	0.245
		4	415284.658	415300.855	443643.116	68.837	-205182.329	429.131	0.246
		5	416731.35	416756.677	452179.423	56.92	-205290.675	435.497	0.253
	50%	2	279195.334	279199.378	293374.563	45.579	-138367.667	405.957	0.22
		3	279875.905	279885.01	301144.749	45.506	-138092.952	417.733	0.233
		4	280694.386	280710.583	309052.844	71.204	-137887.193	423.118	0.239
		5	283824.332	283849.659	319272.405	66.848	-138837.166	426.501	0.243
PSM	10%	2	567471.183	567514.089	613582.497	366.549	-279735.592	367.019	0.181
		3	546752.793	546849.584	615919.764	1007.521	-267376.397	394.482	0.208
		4	9505109.544	505282.073	597332.172	456.046	-244554.772	414.803	0.23
		5	525277.366	525547.664	640555.651	280.498	-252638.683	430.52	0.247
	15%	2	536414.318	536457.224	582525.632	808.557	-264207.159	364.037	0.177
		3	517226.589	517323.379	586393.56	818.516	-252613.294	393.279	0.207
		4	554256.855	554429.384	646479.483	915.612	-269128.428	391.52	0.205
		5	494151.76	494422.057	609430.045	207.825	-237075.88	435.152	0.253
	25%	2	466909.193	466952.099	513020.507	677.334	-229454.596	373.707	0.186
		3	450906.087	451002.878	520073.058	845.518	-219453.044	404.292	0.219
		4	447710.068	447882.597	539932.696	724.47	-215855.034	418.604	0.234
		5	436880.48	437150.778	552158.765	263.332	-208440.24	436.768	0.254
	50%	2	318848.362	318891.268	364959.676	1054.187	-155424.181	350.12	0.164
		3	318741.796	318838.587	387908.767	1036.587	-153370.898	356.1	0.17
		4	327489.436	327661.965	419712.064	2090.985	-155744.718	346.809	0.161
		5	329442.657	329712.954	444720.941	2395.198	-154721.328	354.312	0.168

models extends the methods for solving the student performance prediction problem, framed as a probabilistic sparse tensor factorization problem with the focus on conceptual learning

as the latent factor.

The key challenge of modeling and predicting students' performance lies in the estimation of their states of knowledge while recognizing both the temporal dynamics of knowledge acquisition and the differences in individual learning abilities. Responding to this challenge, the proposed tensor factorization models, PSM and HSM, add constraints to the original MLE problem to capture the knowledge accumulation. PSM features well interpretable parameters, and is shown to compare favorably against two competing tensor factorization methods, SSM and HSM.

Static Student Model that is based on canonical tensor factorization is employed as the baseline model and the state-of-the-art to measure the performance of Homogeneous and Personalized Student Models. Homogeneous student models improves SSM in terms of forcing the model to consider the fact that learning occurs overtime and knowledge is accumulated; however, HSM does not distinguish between different students and assume learning happens at the same rate for all students. Although student specific parameter shows the the model is personalized, the parameter does not contribute in estimating latent factor of knowledge state.

Personalized Student Model offers a new constrained probabilistic tensor factorization model that benefits from the shape of matrices due to the fact that $K << Q, N, T$. PSM personalizes conceptual learning not only for each student but also acknowledges that students may learn different educational contents with different rates. The corresponding optimization problem is attacked using alternating direction method of multipliers which shows satisfactory performance in comparison with block coordinate descent particularly when the quality of obtained solutions matters. The simulation results illustrate that PSM outperforms SSM and HSM and can be used to model student learning dynamics in ITSs. This model can be employed to find the optimal teaching policy and monitor student performance.

Student models developed in this research need to infer many parameters such as $\mathbf{C}, \mathbf{W}, \mathbf{V}, \mathcal{X}$ and so on. As the future direction, using a Bayesian approach based on Markov chain Monte Carlo (MCMC) sampling method can improve the student models in terms of dealing with uncertainties. Although point estimation method of the parameters of interest are computationally less expensive, a fully Bayesian approach to the proposed student models enables one to compute full posterior distributions for $\mathbf{C}, \mathbf{W}, \mathbf{V}, \mathcal{X}$ and compensates the computational complexity with several considerable benefits in the context of learning and content analytics (Lan et al., 2014).

In addition, considering full posterior distributions enables the computation of informative quantities such as credible intervals and posterior modes for all parameters. Moreover, knowing the fact that MCMC methods search the full posterior space, they can escape from local minima. Finally, the hyperparameters used in Bayesian approach generally ease interpretation of model parameters.

Chapter 7

Optimal Teaching Policy

7.1 Introduction

Using an Intelligent Tutoring System (ITS) in a course instruction measurably increases students knowledge gains compared to the same instruction without using an ITS. Although students receiving human one-on-one tutoring services, perform significantly better than their peers in classroom settings (Chi et al., 2011), artificial intelligent-based systems are able to mimic such tutoring behavior and personalize educational interventions for students at scale. Indeed, one-on-one tutoring can be viewed as a sequential decision process where in, at each discrete step, the tutor decides to choose the next activity to take to come up with the best teaching policy customized for each student.

A remarkable number of studies assert that learning is significantly accelerated if the teaching policies are in accordance with the students' need and learning style. In addition, there exists abundant empirical evidence indicating that an interactive dialogue between a tutor and her student is an essential component of learning (Fossati et al., 2015). Therefore, appropriate teaching policies can considerably improve students' performance and make the learning process more effective (Bajraktarevic[1] et al., 2003; Haider et al., 2010; Graf and Viola, 2009; Graf et al., 2008; Liu et al., 2009; Dorça et al., 2013).

In most current ITSs, the interaction occurs when a task is offered by the system and students leave feedback (i.e., their performance on the given task). For instance, a question is selected from an existing question bank and offered to the student. Based on the student's performance on the given question, the ITS makes a decision for the next question(s). However, it is crucial to consider the fact that questions affect student learning gains differently even more, a question may not have the same impact on the student at different times (i.e. questions in early vs. late semester). In addition, questions target different knowledge components and problem solving skills, therefore, an ITS should be able to distinguish effects of all possible actions and select the most influential one.

Similar to one-on-one tutoring, the ITSs behavior can be considered as a sequential decision making where an appropriate action should be chosen from a set of available options at each step. On the other hand, the most desired reward to employ an ITS is student learning gains which are typically observable until the end of training process due to the complex nature of the learning process. It is not straightforward to evaluate students knowledge gains

moment by moment. There is a trade off and between short-term and long-term student learning gains so that many educational interventions that promote short-term performance may not be effective over the long-term. Therefore, in this study, not only the impact of each learning intervention is explored but the trade off between short-term and long-term learning gains is also taken into account.

In spite of the numerous improvements in student interactions with intelligent tutoring systems, there is still a significant gap between human tutoring and ITS. Reducing this gap will require to equip ITSs with models that are able to detect the short-term and long-term gains obtained by each action as well as personalize policies to differences in student abilities and learning styles.

This chapter offers a new framework in student modeling that not only explicitly take the impact of each question into account but it distinguishes also between the gains obtained form correctly and incorrectly answered questions. The proposed model which is founded on top of a constrained tensor factorization model, is employed as a predictive model to optimize teaching policy by finding the optimal sequence of questions personalized to each student. To find personalized optimal teaching policy, models based on Reinforcement Learning (RL) and Model Predictive Control (MPC) are developed.

The rest of this chapter is organized as follows: Section 7.2 reviews state-of-art in prescriptive models in ITSs. Section 7.3 offers a new probabilistic tensor factorization based model to infer students conceptual understanding from their performance data and proposes a fast optimization algorithm to learn the parameters. The optimal teaching problem is formulated as a Reinforcement Problem in Section 7.4 and a Q-learning algorithm is offered to find the optimal sequence of questions. In Section 7.5, a controller based on Model Predictive Control (MPC) is designed to find the optimal sequence of questions in order to maximize knowledge gains. Section 7.6 reports experimental results on synthetic data of varied sizes. Section 7.7 concludes the paper and discusses future research directions.

7.2 Related Work

An integral component in designing ITSs is ability of inferring what a student knows and planing future actions to improve his learning experience. Although substantial research has been done in modeling of student learning and predicting their performance on given activities, there has been significantly less attention on prescriptive models that concentrates on optimizing teaching policy, personalizing learning contents for students and incorporating student interactions with the educational environment (Rafferty et al., 2011). There are a lot of potentials in this line of research and it deserves more attentions.

In an attempt, an automatic, dynamic and probabilistic approach based on reinforcement learning (RL) was proposed to model students learning with emphasizing on the evaluation and comparison of three different strategies for updating the student model during the learning process (Dorça et al., 2013).

In addition, reinforcement learning has been employed to induce two sets of pedagogical policies (i.e., NormGain and InvNormGain) from pre-existing human interaction data. The NormGain set was derived for enhancing tutorial decisions that promote learning while the InvNormGain set was used to enhance with less effective decisions in learning. The two

sets were then tested with students and results illustrated that in the same learning content, different pedagogical policies made a difference in learning and NormGain students performed better than their peers (Chi et al., 2011).

Fossati *et al.* developed an Intelligent Tutoring System called iList that motivates students to learn the concept of linked lists, a challenging topic in data structure. The proposed system is adaptive and provides feedback automatically generated from the history of interaction of students with the system. (Fossati et al., 2015). The impact of immediate and delayed reward functions was investigated on RL-induced policies and the effectiveness of induced policies within an Intelligent Tutoring System called Deep Thought was empirically tested. The author of the study divided students into Fast and Slow learners based on their abilities to solve the problem measured by their average response time on an initial assessment. The outcomes of the study revealed that there was a significant interaction effect between the induced policies and the students inherent abilities so that fast learners are less sensitive to learning settings in that they can learn well regardless of the pedagogical strategies are used; however slow learners more take advantage of from efficient pedagogical strategies than from inefficient ones (Shen and Chi, 2016)

As a sequential decision process, Partially Observable Markov Decision Processes (POMDP) (Sondik, 1978) that is a generalization of a Markov Decision Process also have been reported in ITSs studies. POMDP are very well-known models to deal with situations with incomplete information where the system states are not fully observable. For instance, students' observations such as their performance on given questions does not precisely reflect the conceptual understanding of the materials due to lucky guesses and careless mistakes. POMDPs have been utilized in wide range of practical applications in different domains(see (Lovejoy, 1991)). The main idea in partially observable systems is to exploit memory of the previous observations to reduce ambiguity of a given system (Kaelbling et al., 1998). Although this realistic extension of MDPs offers a more powerful approach to student modeling, the complexity of POMDPs dramatically increases such that finding the optimal teaching policy is computationally intractable Braziunas (2003). All the previous history of observations (i.e., students' performance) and actions (i.e., given questions) are required to obtain the optimal teaching policy.

POMDPs are suitable for determining optimal teaching policy in ITSs since they provide more sophisticated models of learning for the partially observable systems such as student learning. Additionally, POMDP-based models postulate that learners knowledge and learning state cannot be directly captured via observations but they can be approximated by a large number of features. Furthermore, POMDPs consider both the immediate learning reward and the long-term benefit to the student after taking a particular action (i.e., task or question) while more canonical approaches are myopic in terms of maximizing only immediate reward (Rafferty et al., 2011). For the first time, Rafferty et al. used a POMDP approach to model learners knowledge state and obtain individualized teaching policies . Given a learning objective(s) and a model representing the learning process dynamics, POMDPs construct a framework to end up with an optimal teaching policy that optimize the objective(s) (Rafferty et al., 2011).

7.3 Learning-By-Solving Student Model

This section offers a new probabilistic tensor factorization based model that particularly takes question effectiveness into account and explicitly distinguishes impact of questions on student conceptual learning. In addition, the proposed framework presents a personalized student modeling that enables one to incorporate students' specifications to the learning process. Furthermore, it is reasonable to assume that not only each question has its own impact on student learning gains but such effect also relies upon correct or incorrect responses.

7.3.1 Probabilistic Tensor Factorization-based Student Modeling

The core-stone assumption in the proposed model is that the student conceptual understanding evolves for two primary reasons: (i) A student may interact with learning resources (e.g.,textbook, lecture video, lab experiment, run a computer simulation, solve a homework, etc.), all of which are likely to result in an increase of their conceptual understanding. (ii) A student may solve a question given by an ITS platform. It is a simplistic assumption that one presumes all questions have the same effect on student learning. Similar to latent factor models, it is assumed that students needs to learn different K knowledge components in order to conceptually understand the course materials.

Let $\mathcal{X}_{kjt}$ denote student j's state in knowledge component k at time t and $\mathcal{U}_{ijt} = 1(0)$ indicate if the student has answered question i at time t correctly(incorrectly). Assuming Markov property for knowledge gain, student knowledge can be considered as a function of previous knowledge and the question given to the student. Defining such function is consistent with the key assumption about the sources of conceptual understanding. therefore:

$$\mathcal{X}_{kjt} = f(\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}) \tag{7.1}$$

Finding a closed form of (7.1) is a very difficult task specially with sparse data, however, one can replace (7.1) with a linear system that can approximate relation between current state of knowledge and previous knowledge as well as correct/incorrect responses.

$$\mathcal{X}_{kjt} = \sum_{l=1}^{K} H_{lk} A_{kj} \mathcal{X}_{ljt-1} + \mathcal{B}^{+}_{kit-1} \mathcal{U}_{ijt-1} + \mathcal{B}^{-}_{kit-1} \Big(1 - \mathcal{U}_{ijt-1}\Big) + \epsilon, \tag{7.2}$$

where $\mathbf{H}$ is a $K \times K$ matrix capturing the interconnection between knowledge components such that $H_{ll} = 1 \quad \forall \quad l = 1, \ldots, K$ and $0 \leq H_{lk} < 1 \quad \forall \quad l \neq k$ while $\mathbf{H} = \mathbf{I}_K$ means that all knowledge components are independent from each other. Student j learning rate in knowledge component k is represented with A_{kj}, where $0 < A_{kj} < 1$. Let $\mathcal{B}^{+}_{kit}$ and $\mathcal{B}^{-}_{kit}$ denote how question i helps one to learn knowledge k at time t if question is answered correctly and incorrectly, respectively. One can define binary variable $\mathcal{U}_{ijt} = 1$ if student j answers question i at time t correctly and $\mathcal{U}_{ijt} = 0$ otherwise. However, the uncertainty of this system is captured by $\epsilon \sim \mathcal{N}(0, \sigma^2_{kj})$, therefore, the knowledge state of each student is assumed to be a random variable with a Gaussian distribution:

$$\mathcal{X}_{kjt} | \mathcal{X}_{kjt-1}, \mathcal{U}_{ijt} \sim \mathcal{N}\Big(\mathcal{M}_{kjt}, \sigma^2_{kj}\Big), \tag{7.3}$$

where

$$\mathcal{M}_{kjt} = E[\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}] = \sum_{l=1}^{K} H_{lk} A_{kj} \mathcal{X}_{ljt-1} + \mathcal{B}^{+}_{kit-1} \mathcal{U}_{ijt-1} + \mathcal{B}^{-}_{kit-1} \Big(1 - \mathcal{U}_{ijt-1}\Big), \quad (7.4)$$

is the mean of the distribution. Defining $\mathcal{T}_{ijt}$ as

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ik} \mathcal{M}_{kjt} + \theta_j - d_i \quad \forall \quad i = 1, \ldots, Q \;\; j = 1, \ldots, N, \; t = 1 \ldots T, \quad (7.5)$$

the probability of the observation is:

$$\mathcal{Y}_{ijt} \sim Ber(\Phi(\mathcal{T}_{ijt})),$$

where d_i, and θ_j denote the difficulty of question i and student j specific parameter. *Ber(p)* shows a Bernoulli distribution with success probability p and $\Phi(x)$ is a logit function. The probability mass function of $\mathcal{Y}_{ijt}$ given the parameters is defined as:

$$P_{\mathbf{Y}}(\mathcal{Y}_{ijt}|\mathcal{X}_{:jt}, \theta_j, d_i) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}} [1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}} \quad (7.6)$$

Aster observing students performance on given question at different time,Ω_{obs}, the parameters of (7.6) should be estimated. Therefore, the following likelihood maximization problem needs to be solved to obtain the parameters of the model.

$$(P4): \max_{\mathcal{B}^{+}, \mathcal{B}^{-}, \mathcal{X}, \mathbf{H}, \mathbf{W}, \boldsymbol{\theta}, \mathbf{d}} \sum_{(i,j,t) \in \Omega_{obs}} log(P_{\mathbf{Y}}(\mathcal{Y}_{ijt}|\mathcal{X}_{:jt}, d_i, \theta_j))$$

$$s.t.$$

$$\|\mathcal{B}^{+}_{:it}\|_1 \leq \delta^{+} \;\; \forall \;\; i = 1, \ldots, Q, \quad t = 1, \ldots, T \quad (7.7)$$

$$\|\mathcal{B}^{-}_{:it}\|_1 \leq \delta^{-} \;\; \forall \;\; i = 1, \ldots, Q, \quad t = 1, \ldots, T \quad (7.8)$$

$$\|\mathcal{X}_{::t}\|_F = \xi \;\; \forall \;\; t = 1, \ldots, T \quad (7.9)$$

$$\mathcal{M}_{kjt} \leq \mathcal{M}_{kjt+1} \; \forall \;\; k = 1, \ldots, K, \;\; j = 1, \ldots, N, \; t = 0, \ldots, T-1 \quad (7.10)$$

$$0 \leq H_{lk} < 1 \; \forall \quad l = 1, \ldots, K, \;\; k = 1, \ldots, K, \quad (7.11)$$

$$\epsilon \leq A_{kj} \leq 1 - \epsilon \; \forall \quad K = 1, \ldots, K, \;\; j = 1, \ldots, N, \quad (7.12)$$

$$\mathcal{B}^{+}_{kit}, \mathcal{B}^{-}_{kit} \geq 0 \; \forall \quad i = 1, \ldots, Q, \;\; k = 1, \ldots, K, \; t = 1, \ldots, T, \quad (7.13)$$

$$W_{ik} \geq 0 \; \forall \;\; k = 1, \ldots, K, \;\; i = 1, \ldots, Q \quad (7.14)$$

This constrained optimization problem can be simplified by relaxing some constraints. The optimization model using Lagrangian multipliers is defined as (7.15) :

$$\begin{aligned}(P_4') \min_{\mathcal{B}^{+}, \mathcal{B}^{-}, \mathcal{X}, \mathbf{H}, , \mathbf{W}, \boldsymbol{\theta}, \mathbf{d}} \Gamma = & \sum_{(i,j,t) \in \Omega_{obs}} -log(P_{\mathbf{Y}}(\mathcal{Y}_{ijt}|\mathcal{X}_{:jt}, d_i, \theta_j)) + \lambda_0 \sum_{i=1}^{Q} \|\mathbf{w}_i\|_1 \\ & + \lambda_1 \sum_{i=1}^{Q} \sum_{t=1}^{T} \|\mathcal{B}^{+}_{:it}\|_1 + \frac{\lambda_2}{2} \sum_{i=1}^{Q} \sum_{t=1}^{T} \|\mathcal{B}^{-}_{:it}\|_1 + \frac{\lambda_3}{2} \sum_{t=1}^{T} \|\mathcal{X}_{::t}\|_F \end{aligned} \quad (7.15)$$

$$
\begin{aligned}
&st: \\
&\mathcal{M}_{kjt} \leq \mathcal{M}_{kjt+1} \; \forall \;\; k = 1, \ldots, K, \;\; j = 1, \ldots, N, \; t = 0, \ldots, T-1 && (7.16)\\
&\mathcal{B}^{+}_{kit}, \mathcal{B}^{-}_{kit} \geq 0 \; \forall \;\; i = 1, \ldots, Q, \;\; K = 1, \ldots, K, \; t = 1, \ldots, T && (7.17)\\
&W_{ik} \geq 0 \; \forall \;\; k = 1, \ldots, K, \;\; i = 1, \ldots, Q && (7.18)\\
&0 \leq H_{lk} < 1 \; \forall \;\; l = 1, \ldots, K, \;\; k = 1, \ldots, K, && (7.19)\\
&\epsilon \leq A_{kj} \leq 1 - \epsilon \; \forall \;\; K = 1, \ldots, K, \;\; j = 1, \ldots, N, && (7.20)
\end{aligned}
$$

Constraints (7.16) to (7.20) are satisfied using projection methods. Gradients of the likelihood function respect to the variables are calculated as:

$$\nabla\Gamma_{\mathcal{X}_{kjt}} = -A_{kj}\Big[\sum_{l=1}^{K} H_{lk}\Big[\sum_{i=1}^{Q}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}})\Big]\Big] + \lambda_3 \sum_{t=1}^{T} \frac{\mathcal{X}_{kjt}}{\|\mathcal{X}_{::t}\|_F} \tag{7.21}$$

$$\nabla\Gamma_{H_{lk}} = -\sum_{j=1}^{N} A_{kj}\Big[\sum_{i=1}^{Q}\sum_{t=1}^{T} \mathcal{X}_{ljt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}})\Big] \tag{7.22}$$

$$\nabla\Gamma_{\mathcal{B}^{+}_{kit}} = -\sum_{j=1}^{N} \mathcal{U}_{pjt-1}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) \tag{7.23}$$

$$\nabla\Gamma_{\mathcal{B}^{-}_{kit}} = \sum_{j=1}^{N} (\mathcal{U}_{pjt-1} - 1)(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) \tag{7.24}$$

$$\nabla\Gamma_{A_{kj}} = -\sum_{l=1}^{K} H_{lk}\Big[\sum_{t=1}^{T} \mathcal{X}_{ljt}\Big[\sum_{i=1}^{Q}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}})\Big]\Big] \tag{7.25}$$

$$\nabla\Gamma_{W_{ki}} = -\sum_{j=1}^{N}\sum_{t=1}^{T} \mathcal{X}_{kjt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) \tag{7.26}$$

The gradient information can be used in the optimization procedure. A fast version of Block Coordinate Descent algorithm is adapted to attack the optimization problem in (7.15)

7.3.2 Accelerated Randomized Block Coordinate Descent

Coordinate Descent-based methods are among popular methods originally developed for solving smooth unconstrained minimization problems. The main privilege of these schemes is the simplicity of each iteration such that updating of variables and finding the search direction are computationally inexpensive. Nevertheless, coordinate descent methods are criticized in several aspects such as lack of theoretical justification of convergence (Nesterov, 2012). Nesterov presented a fast version of coordinate descent for large-scale optimization which is closely related to the accelerated full gradient method that he proposed in 1983 and have become extremely popular in recent years (Nesterov, 2012; Wright, 2015).

The idea behind the fast representation of coordinate descent is the use of momentum in selecting the step-size in the search direction that combines new gradient information with the previous search direction. It is worth mentioning that the proposed method is related

to other gradient-based techniques such as conjugate gradient method and the heavy-ball method (see Polyak (1987)).

The first step of this method is to calculate the step-sizes α_k and β_k at iteration k of the algorithm.

$$\alpha_k = \frac{n - \gamma_k \sigma}{\gamma_k (n^2 - \sigma)}, \quad \beta_k = 1 - \frac{\gamma_k \sigma}{n},$$

where n is the number of variables needs to be updated and σ is constant and in the strongly convex case $\sigma > 0$. In addition, γ_k is the largest root obtained from solving the following quadratic equation:

$$\gamma_k^2 - \frac{\gamma_k}{n} = (1 - \frac{\gamma_k \sigma}{n})\gamma_{k-1}^2$$

The search direction is defined as the combination of current solution x_k and v_k defined below:

$$y_k = \alpha_k v_k + (1 - \alpha_k) x_k.$$

Defining $\nabla f(x_k)$ as the gradient of x_k and x_{k+1} is updated as follows:

$$x_{k+1} = y_k - \frac{1}{L_M} \nabla f(x_k),$$

where L_M is coordinate Lipschitz constant. It should be noted that one can find coordinate Lipschitz constant for convex functions (Wright, 2015), however, finding coordinate Lipschitz constant is not straightforward. v_{k+1} is updated as follows:

$$v_{k+1} = \beta_k v_k - (1 - \beta_k) y_k - (\frac{\gamma_k}{L_M}) \nabla f(x_k)$$

Algorithm 7 explains how Nesterov can be used to accelerate the randomized block coordinate descent.

7.4 Reinforcement Learning Approach

To find the optimal sequence of question, i.e., optimal teaching policy, one needs be able to deal with high-dimensional and uncertain states in student's knowledge and questions in order to personalize the optimal sequence of questions for each students in a highly complex environment. Inferring the state of an agent from noisy, partially observations has been a popular subject of research in the operation research, artificial intelligence and machine learning communities. In this section, two sequential decision making processes are implemented to find personalized, optimal teaching policy for each student. The first approach is based on Reinforcement learning that makes use of Markov Decision Process (MDP) as the core-stone model to find the optimal sequence of actions. The second approach which is more realistic in the student modeling application is Partially Observable Markov Decision Process (POMDP) which extends MDP to the situation that conceptual understating of students in inferred from the incomplete observation reflection lucky guesses and careless mistakes. However, both approaches need to have transition probability matrices which are obtained from model IV.

Algorithm 7 Accelerated Randomized Block Coordinate Descent Algorithm

Input: Observed Tensor $\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$.
Output: Completed Tensor $\mathcal{Y}$, Matrices $\mathcal{X}$, $\mathcal{B}^+$ and $\mathcal{B}^-$ $\mathbf{A}$, $\mathbf{W}$ and $\mathbf{H}$.
LBS_Model()

1. Initialize randomly $\mathcal{X}$, $\mathcal{B}^+$, $\mathcal{B}^-$,$\mathbf{A}$, $\mathbf{W}$ and $\mathbf{H}$.
2. set $V^0_{\mathcal{X}} \leftarrow \mathcal{X}$, $V^0_{\mathcal{B}^+} \leftarrow \mathcal{B}^+$, $V^0_{\mathcal{B}^-} \leftarrow \mathcal{B}^-$, $k \leftarrow 1$ and $\gamma_0 \leftarrow 0$
3. **while** (stopping criteria) **do**
4. choose γ_k to be the larger root of $\gamma_k^2 - \frac{\gamma_k}{n} = (1 - \frac{\gamma_k \sigma}{n})\gamma_{k-1}^2$.
5. set $\alpha_k = \frac{n - \gamma_k \sigma}{\gamma_k (n^2 - \sigma)}$ and $\beta_k = 1 - \frac{\gamma_k \sigma}{n}$
6. randomly select $t \in \{1, \ldots, T\}$
7. $\mathbf{Y}_{\mathcal{X}} \leftarrow \alpha_k \mathbf{V}^k_{\mathcal{X}} + (1 - \alpha_k)\mathcal{X}_{t::}$, $\mathbf{Y}_{\mathcal{B}^+} \leftarrow \alpha_k \mathbf{V}^k_{\mathcal{B}^+} + (1 - \alpha_k)\mathcal{B}^+_{t::}$, $\mathbf{Y}_{\mathcal{B}^-} \leftarrow \alpha_k \mathbf{V}^k_{\mathcal{B}^-} + (1 - \alpha_k)\mathcal{B}^-_{t::}$
8. calculate $\nabla\Gamma_{\mathcal{X}_{t::}}$, $\nabla\Gamma_{\mathcal{B}^+_{t::}}$, $\nabla\Gamma_{\mathcal{B}^-_{t::}}$, $\nabla\Gamma_{d_i}$, $\nabla\Gamma_{\mathbf{H}}$ and $\nabla\Gamma_{\theta_j}$ using (7.21)-(7.24)
9. update $\mathcal{X}_{t::} \leftarrow \max\left\{\mathcal{X}_{t-1::}, \mathbf{Y}_{\mathcal{X}} - \frac{1}{L_M}\nabla\Gamma_{\mathcal{X}_{t::}}\right\}$
10. update $\mathcal{B}^+_{t::} \leftarrow \max\left\{\mathbf{0}, \mathbf{Y}_{\mathcal{B}^+} - \frac{1}{L_M}\nabla\Gamma_{\mathcal{B}^+_{t::}}\right\}$ and $\mathcal{B}^-_{t::} \leftarrow \max\left\{\mathbf{0}, \mathbf{Y}_{\mathcal{B}^-} - \frac{1}{L_M}\nabla\Gamma_{\mathcal{B}^-_{t::}}\right\}$
11. update $\mathbf{H} \leftarrow \max\left\{\mathbf{0}, \min\{\mathbf{1}, \mathbf{H} - \beta\nabla\Gamma_{\mathbf{H}}\}\right\}$
12. update $\mathbf{A} \leftarrow \max\left\{\mathbf{0}, \min\{\mathbf{1}, \mathbf{A} - \beta\nabla\Gamma_{\mathbf{A}}\}\right\}$
13. update $\mathbf{W} \leftarrow \max\left\{\mathbf{0}, \mathbf{W} - \beta\nabla\Gamma_{\mathbf{W}}\right\}$
14. update $d_i^{new} \leftarrow d_i^{old} - \beta\nabla\Gamma_{d_i}$ and $\theta_j^{new} \leftarrow \theta_j^{old} - \beta\nabla\Gamma_{\theta_j}$
15. update $\mathbf{V}^{k+1}_{\mathcal{X}} \leftarrow \beta_k \mathbf{V}^{k+1}_{\mathcal{X}} + (1 - \beta_k)\mathbf{Y}_{\mathcal{X}} - \frac{\gamma_k}{L_M}\nabla\Gamma_{\mathcal{X}_{t::}}$
16. update $\mathbf{V}^{k+1}_{\mathcal{B}^-} \leftarrow \beta_k \mathbf{V}^{k+1}_{\mathcal{B}^+} + (1 - \beta_k)\mathbf{Y}_{\mathcal{B}^+} - \frac{\gamma_k}{L_M}\nabla\Gamma_{\mathcal{B}^+_{t::}}$, $\mathbf{V}^{k+1}_{\mathcal{B}^-} \leftarrow \beta_k \mathbf{V}^{k+1}_{\mathcal{B}^-} + (1 - \beta_k)\mathbf{Y}_{\mathcal{B}^-} - \frac{\gamma_k}{L_M}\nabla\Gamma_{\mathcal{B}^-_{t::}}$
17. calculate objective function using (7.15)
18. **end**
19. report AIC, AIC_c, BIC, MSE

7.4.1 Reinforcement Learning Formulation

To find the optimal teaching policy and tailor it for each student, a probabilistic model is needed to capture the dynamics of learning and the impact of each question on students' conceptual understanding. Solving model IV helps to extract the learning dynamics of students and provide the required components for finding the optimal policy.

Let $\mathcal{S}_{kjt}$ denote student j's state in knowledge component k at time t such that $\mathcal{S}_{kjt} \in \mathbf{S}$ (i.e., $\mathcal{S}_{kjt}$ can be considered as a discrete or continuous variable) and $\mathcal{A}_{ijt} \in \mathbf{A}$ represent question i is given to student j at time t where $\mathbf{S}$ and $\mathbf{A}$ indicate the state and action spaces, respectively. Initial state is indicated by $\mathcal{S}^0$ with probability $P(\mathcal{S}_{::0})$ and $P(\mathcal{S}_{kjt}|\mathcal{S}_{kjt-1}, \mathcal{A}_{ijt})$ denotes the transition probability (one can think of (7.3) as the transition probability and use P_4' to estimate the parameters). The reward function is denoted with $r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt})$ showing the reward of taking question i when student j is in state $\mathcal{S}_{kjt}$ at time t and defined as how much improvement is obtained in the knowledge state. It also considers more bonus for trying more difficult questions.

$$r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) = \beta \Delta \mathcal{S}_{kjt} + d_i, \tag{7.27}$$

where $\Delta \mathcal{S}_{kjt} = \mathcal{S}_{kjt} - \mathcal{S}_{kjt-1}$ and d_i captures the difficulty of question i. The return of an episode in the MDP is the discounted sum of rewards received by the agent during that episode:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) \tag{7.28}$$

where R_t is the discounted sum of rewards and $\gamma \in [0, 1]$ is a discount factor for future rewards. The goal of Markov Decision Process (MDP) model is to find a teaching policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which maximizes the expected reward over all time slots, $E[R_t]\ \forall\ t$. This is an combinatorial optimization problem with $Q!$ possible actions. The state-action value function is defined as the expected return starting from a given state $\mathcal{S}_{kjt}$ and taking an action $\mathcal{A}_{ijt}$ following π. Q^π is expressed in a recursive way using the Bellman equation:

$$Q_\pi(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) = r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) + \gamma \sum_{\mathcal{S}_{kjt+1}} P(\mathcal{S}_{kjt+1}|\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) Q_\pi\Big(\mathcal{S}_{kjt+1}, \pi(\mathcal{S}_{kjt+1})\Big) \tag{7.29}$$

There are two primary families of policies often used in MDP: value-based, and actor-based policies. For value-based policies, the policys decisions are directly conditioned on the value function. To find the optimal teaching policy, one should solve the following problem for all combinations of states and actions.

$$\pi_Q(\mathcal{S}) = \arg \max_{\mathcal{A}_{ijt} \in \mathcal{A}} Q(\mathcal{S}_{kjt+1}, \mathcal{A}_{ijt+1}) \tag{7.30}$$

Q-learning is a popular method for solving reinforcement learning problems. Algorithm 8 explains Q-learning method to find optimal teaching policy.

Algorithm 8 Q-learning for finding optimal teaching policy

Input: $P(\mathcal{S}_{kjt}|\mathcal{S}_{kjt-1}, \mathcal{A}_{ijt})$ and $r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt})$.
Output: Optimal Policy $\pi^*_Q(\mathcal{S})$.
Q-Learning()

1. $t \leftarrow 0$
2. Initialize $V_t(\mathcal{S}) = 0 \; \forall \; \mathcal{S}_{kjt} \in \mathbf{S}$, $\mathbf{A}^+ = \mathbf{A}$
3. **do**
4. $t \leftarrow t + 1$
5. **for** all states $\mathcal{S}_{kjt} \in \mathbf{S}$
6. **for** all actions $\mathcal{A}_{ijt} \in \mathbf{A}^+$
7. $Q_\pi(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) = r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) + \gamma \sum_{\mathcal{S}_{kjt+1}} P(\mathcal{S}_{kjt+1}|\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) Q_\pi \Big(\mathcal{S}_{kjt+1}, \pi(\mathcal{S}_{kjt+1}) \Big)$
8. $V_t(\mathcal{S}) = \max_{\mathcal{A}_{ijt} \in \mathcal{A}} Q(\mathcal{S}_{kjt+1}, \mathcal{A}_{ijt+1})$
9. $\pi_Q(\mathcal{S}) = \arg\max_{\mathcal{A}_{ijt} \in \mathcal{A}} Q(\mathcal{S}_{kjt+1}, \mathcal{A}_{ijt+1})$
10. $\mathbf{A}^+ \leftarrow \mathbf{A}^+ - \{\mathcal{A}^*_{ijt}\}$
11. **while** $\|V_t(\mathcal{S}) - V_{t-1}(\mathcal{S})\| \leq \epsilon$
12. return $\pi^*_Q(\mathcal{S})$

7.4.2 Challenges of Reinforcement Learning Approach

The main challenge of using reinforcement model is the fact that in this application, each action can be effectively used once. If a student is given a particular question, this question cannot significantly boost the student learning gain in the second time. One can assume that the student may forget what the question has taught her, however, this assumption makes the problem much more difficult.

The consequence of considering the fact that there is only one chance for each action to be selected results in a combinatorial optimization problem which makes reinforcement learning approach ineffective in real time application such.

7.5 Constrained Model Predictive Control Approach

To find the best sequence of question known as optimal teaching policy and personalize the solution for each student, a Constrained Model Predictive Control (CMPC) is utilized to find personalized optimal teaching policy in a real-time way. The predictive model in MPC is the student-question interaction model that is updated in each iteration of MPC and estimate the parameters based on the new observations. The MPC optimizer find the new optimal policy from using available questions.

The core-stone of CMPC is that the current control action is obtained by solving a finite horizon open-loop optimal control problem at each sampling instant. The current state of the plant (i.e. students) is considered as the initial state and the optimization finds an optimal control sequence and offers the first control in this sequence to be applied to the students in the course. An important advantage of MPC for this problem is its ability and flexibility to deal with hard constraints on controls (i.e. questions) and states (students knowledge). Due to MPC privileges, therefore, it has widely been applied in petro-chemical and related industries where considering constraints is crucial (Mayne et al., 2000).

7.5.1 Student Learning Dynamics

The key-stone in many control systems is the underlying dynamics that represents behaviors of the system. It is assumed that the student learning dynamics is captured using (7.2). Let $x_{kj}(t)$ denote student j state in knowledge component k at time t and $\widehat{A}_{kj}$, $\widehat{H}_{kl}$, $\widehat{B}^{+}_{kit}$,$\widehat{B}^{-}_{kit}$ are estimated parameters obtained from model $P_4{}'$ and $\widehat{P}_{ijt} \in [0,1]$ is the estimated probability that performance of student j on problem i at time t.

$$\widehat{P}_{ijt} = P(\mathcal{Y}_{ijt} = 1 | \mathcal{X}_{:jt}, \theta_j, d_i).$$

The student learning dynamics is explained as follows:

$$x_{kj}(t+1) = \sum_{l=1}^{K} \widehat{H}_{kl}\widehat{A}_{kj}x_{kj}(t) + \left[\widehat{B}^{+}_{kit}\widehat{P}_{ijt} + \widehat{B}^{-}_{kit}(1-\widehat{P}_{ijt})\right]u_{ij}(t), \tag{7.31}$$

where $x_{kj}(t) \in \mathbf{R}$ and $u_{ij}(t) = 1(0)$ is the control command defined as whether question i is given to student j at time t. The main difference between (7.2) with (7.31) is that

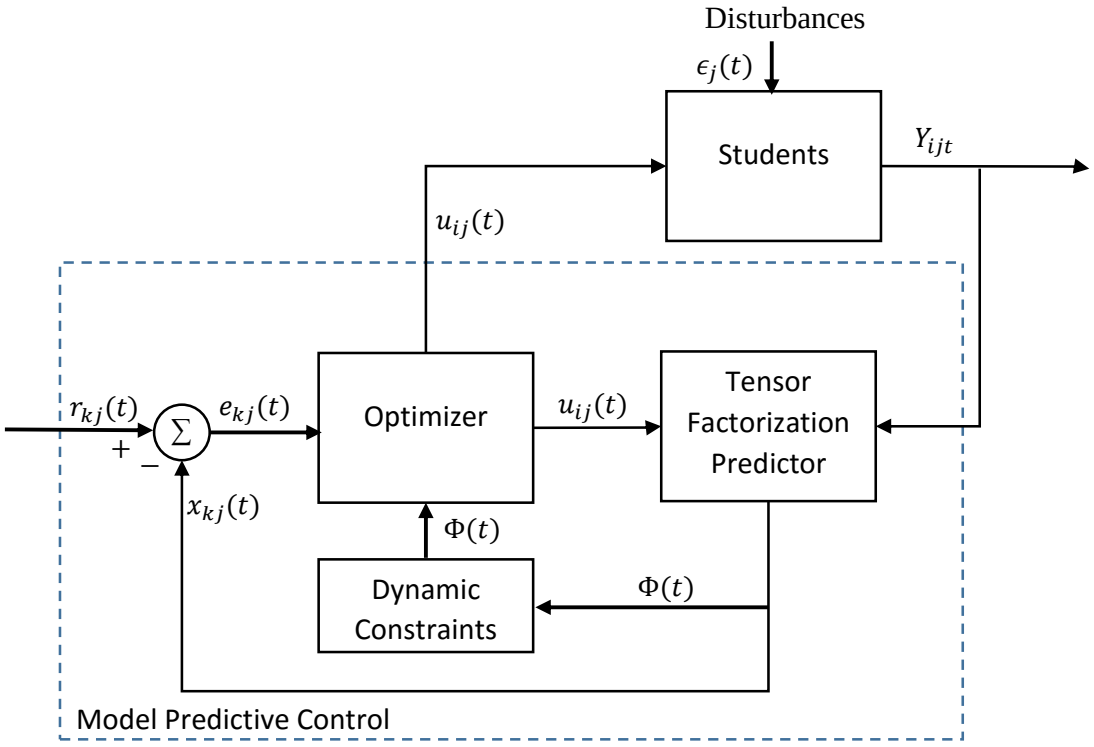

Figure 7.1: Constrained Model Predictive Control Diagram.

actualization of student performance is not available, therefore, the estimated probabilities are used instead of actual values. The objective of the controller is to find the optimal input for this system (i.e. $u_{ij}(t)$) by means of minimizing the cost function in (7.32).

$$J = \sum_{\tau=t}^{T} \gamma^{\tau} \Big[\sum_{k=1}^{K} [r_{kj}(\tau) - x_{kj}(\tau)]^2 \Big], \tag{7.32}$$

where $r_{kj}(t)$ represent the desired state of student j in knowledge component k at time t and γ, $0 \leq \gamma \leq 1$, is a discount factor assigning more weights to the near future to accelerate student learning. Considering the reference (i.e. the desired state) is not necessary in this CMPC, however, it enables the instructor to provide a specific learning trajectory for students. Assuming the time is discrete, the planing horizon is $T - t$ and the controller find the best sequence of question considering all predictions in this period. Figure 7.1 depicts different components of CMPC.

7.5.2 CMPC Optimization Model

After estimating parameters in (7.31), the next step is to find the optimal sequence of questions that is personalized to students for the next $T - t$ steps ahead. The challenging part of this problem is that each question has only one chance to be selected. When a question is given to the student for the second time, it is not as effective as the first time because student already knows the solution. The following model predictive model optimizes the sequence of questions that student j should be given from time t to T based on predictions

at time t.

$$(P_5) \quad \min_{\mathbf{u}} J = \sum_{\tau=t}^{T} \gamma^{\tau} \Big(\sum_{k=1}^{K} [r_{kj}(\tau) - x_{kj}(\tau)]^2 \Big) + M \sum_{\tau=0}^{t-1} \sum_{i \in \Phi(\tau)} u_{ij}(\tau) \tag{7.33}$$

$st:$

$$x_{kj}(\tau+1) = \widehat{A}_{kj} \sum_{l=1}^{K} \widehat{H}_{kl} x_{lj}(\tau) + \sum_{i=1}^{Q} \Big(\widehat{B}^{+}_{ki\tau} \widehat{P}_{ij\tau} + \widehat{B}^{-}_{ki\tau} (1 - \widehat{P}_{ij\tau}) \Big) u_{ij}(\tau) \; \forall \; \tau, k \tag{7.34}$$

$$\sum_{i=1}^{Q} u_{ij}(\tau) = 1 \;\; \forall \;\; \tau = t, \ldots, T \tag{7.35}$$

$$\sum_{\tau=t}^{T} u_{ij}(\tau) \leq 1 \;\; \forall \;\; i = 1, \ldots, Q \tag{7.36}$$

$$x_{kj}(t-1) = \widehat{x}_{kj}(t-1) \;\; \forall \;\; k = 1, \ldots, K \tag{7.37}$$

$$u_{ij}(\tau) \in \{0, 1\} \;\; \forall \;\; \tau = t, \ldots, T, \;\; i = 1, \ldots, Q \tag{7.38}$$

In this model, the objective function is quadratic and M is a penalty for those questions that have been used. Constraints 7.40 indicate the student learning dynamics for all knowledge components. Since it is not clear which question will be selected, one needs to sum over all possible questions in 7.40. Constraints 7.41 and 7.42 guarantee that one question is selected at each time step and no question is chosen more than once. Finally, $\widehat{x}_{kj}(t-1)$ shows the estimation of student j knowledge at time $t-1$ as the initial state of the system.

However, the optimization model is a combinatorial optimization problem with quadratic function. Solving such optimization problem is as hard as quadratic assignment problem. In order to have a more efficient algorithm, one can reduce the complexity of this optimization problem while all the key features of CMPC are preserved. Since the objective of CMPC is to boost students knowledge over time, one can ignore $r_{kj}(t)$ and assume that the goal is to maximize all students knowledge states. As a result, one can replace the objective function with a linear function which extremely makes the optimization more straightforward while the main functionality of the controller is held.

$$(P_6) \quad \min_{\mathbf{u}} J = -\sum_{\tau=t}^{T} \gamma^{\tau} \sum_{k=1}^{K} x_{kj}(\tau) + M \sum_{\tau=0}^{t-1} \sum_{i \in \Phi(\tau)} u_{ij}(\tau) \tag{7.39}$$

$st:$

$$x_{kj}(\tau+1) = \widehat{A}_{kj} \sum_{l=1}^{K} \widehat{H}_{kl} x_{lj}(\tau) + \sum_{i=1}^{Q} \Big(\widehat{B}^{+}_{ki\tau} \widehat{P}_{ij\tau} + \widehat{B}^{-}_{ki\tau} (1 - \widehat{P}_{ij\tau}) \Big) u_{ij}(\tau) \; \forall \; \tau, k \tag{7.40}$$

$$\sum_{i=1}^{Q} u_{ij}(\tau) = 1 \;\; \forall \;\; \tau = t, \ldots, T \tag{7.41}$$

$$\sum_{\tau=t}^{T} u_{ij}(\tau) \leq 1 \;\; \forall \;\; i = 1, \ldots, Q \tag{7.42}$$

$$x_{kj}(t-1) = \widehat{x}_{kj}(t-1) \;\; \forall \;\; k = 1, \ldots, K \tag{7.43}$$

$$u_{ij}(\tau) \in \{0,1\} \ \ \forall \ \ \tau = t, \ldots, T, \ \ i = 1, \ldots, Q \tag{7.44}$$

Algorithm 9 CMPC optimal teaching policy

Input: Observed Tensor $\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$.
Output: The optimal teaching policy, $\mathcal{U}^*$
CMPC()

1. $t \leftarrow 0$
2. $\Phi(0) \leftarrow$ randomly assign the first set of questions to students at $t = 0$
3. **while** $(t \leq T - 1)$
4. update $\mathcal{Y}_{::t}$
5. **for** all students
6. $\mathcal{P}_{:jt}, \mathcal{B}^{+}{}_{:jt}, \mathcal{B}^{-}{}_{:jt}, \mathbf{H}, \mathbf{A}_{:j} \leftarrow$ LBS_Model($\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$) /* algorithm 7 */
7. $\mathcal{U}^*_{:jt} \leftarrow$ optimize_cmpc($\mathcal{P}_{:jt}, \mathcal{B}^{+}{}_{:jt}, \mathcal{B}^{-}{}_{:jt}, \mathbf{H}, \mathbf{A}_{:j}$) /* an IP algorithm to solve (P_6) */
8. give $\mathcal{U}^*_{:jt}$ to the student to work on it.
9. $\Phi(t+1) \leftarrow \Phi(t) \cup \mathcal{U}^*_{:jt}$
10. $t \leftarrow t + 1$
11. return $\mathcal{U}^*$

7.6 Experimental Studies

This section describes the process of experiments setup and implementation of algorithm. As discussed before, there are several datasets that researchers in this area use to test their algorithms, however, one needs the data representing the state of knowledge of students to be able to measure the algorithm performance and interpretability of the models. Therefore, the synthetic data that is generated based on Kalman filter method is employed. To test both accuracy and scalability of the proposed model, three classes of instances are considered: 1) Small-sized problems 2)Medium-sized problems 3)Large- sized problems. It is worth emphasizing that the size of instances depends on K, Q, N and T. First of all, the performance of LBS student model is tested via different instances with four levels of sparsity. Table 7.1 summarizes the experiment plan.

7.6.1 Implementation

To implement the ARBCD algorithm, the following aspects were used:

Table 7.1: Experiments setup for testing Tensor Factorization Models.

Category	Experiment ID	Question	Student	Time (week)	KC	Sparsity	Estimated KC	Iterations	Num of Run
Small	1	100	20	6	3	25%	2-5	500	30
	2	100	20	6	3	50%	2-5	500	30
	3	100	20	6	3	75%	2-5	500	30
	4	100	20	6	3	85%	2-5	500	30
Medium	5	200	40	10	5	15%	2-5	200	25
	6	200	40	10	5	25%	2-5	200	25
	7	3200	40	10	5	50%	2-5	200	25
	8	200	40	10	5	75%	2-5	200	25
Large	9	500	100	15	8	10%	2-5	100	20
	10	500	100	15	8	15%	2-5	100	20
	11	500	100	15	8	25%	2-5	100	20
	12	500	100	15	8	50%	2-5	100	20

- **Software:** We used a custom code written Python using NumPy, Grubi and Pandas. Double precision values were used for matrices and tensors. The observation matrix $\mathcal{Y}$ is binary and highly sparse so the compressed column and row representations is recommended.

- **Hardware:** Different hardware for various experimental setup was used. For experiments 4 to 8, we used a desktop with Intel(R) Core(TM)i5 3GHz processor, 8GB RAM and 64 bit operating system. For experiments 9 to 12, Dell machines with 12x2.40GHz Intel Xeon E5645 Processor Cores and 12 cores per node with 48 GB memory and Linux (RedHat Enterprise Linux 6.1 2.6.32 Kernel) operating systems were used.

- **Reporting:** The main challenge in dealing with high dimension non-convex optimization problems is the huge number of local optima causing the majority iteration-based algorithms get stuck and not converge to the global optimum solution. Anther issue with such large-scale optimization problems is that the optimal solution is usually unknown particularly in real-world applications. Consequently, defining a stopping criterion is problematic even though one knows the optimal solution. In this paper, we consider number of iterations as the stopping criteria. With the number of iterations, we fix the number of iterations and ARBCD reports the best found solution. To ensure that random initial conditions do not affect the results, each experiment is run several times for each experiment. In addition, the time to find the best solution from initialization is captured and reported. In order to provide better insights about the model and help one to compare the models, criteria such as Akaike Information Criteria (AIC), AIC corrected (AIC_c), Bayesian Information Criteria (BIC), Log likelihood, Frobenius Norm and Mean Square Error (MSE) described below are reported.

$$AIC = 2k - 2\Gamma, \quad AIC_c = AIC + \frac{2k(k+1)}{n-k-1}, \quad BIC = -2\Gamma + k\log(n),$$

where k is the number of parameters, Γ denotes the loglikelihood function and n indicates the sample size. In this study, Frobenius Norm is used as below.

$$F = \sum_{t=1}^{T} \|P_{::t} - \widehat{P}_{::t}\|_F = \sum_{t=1}^{T} \left(\sum_{i=1}^{Q} \sum_{j=1}^{N} |p_{ijt} - \widehat{p}_{ijt}|^2 \right)^{\frac{1}{2}}, \tag{7.45}$$

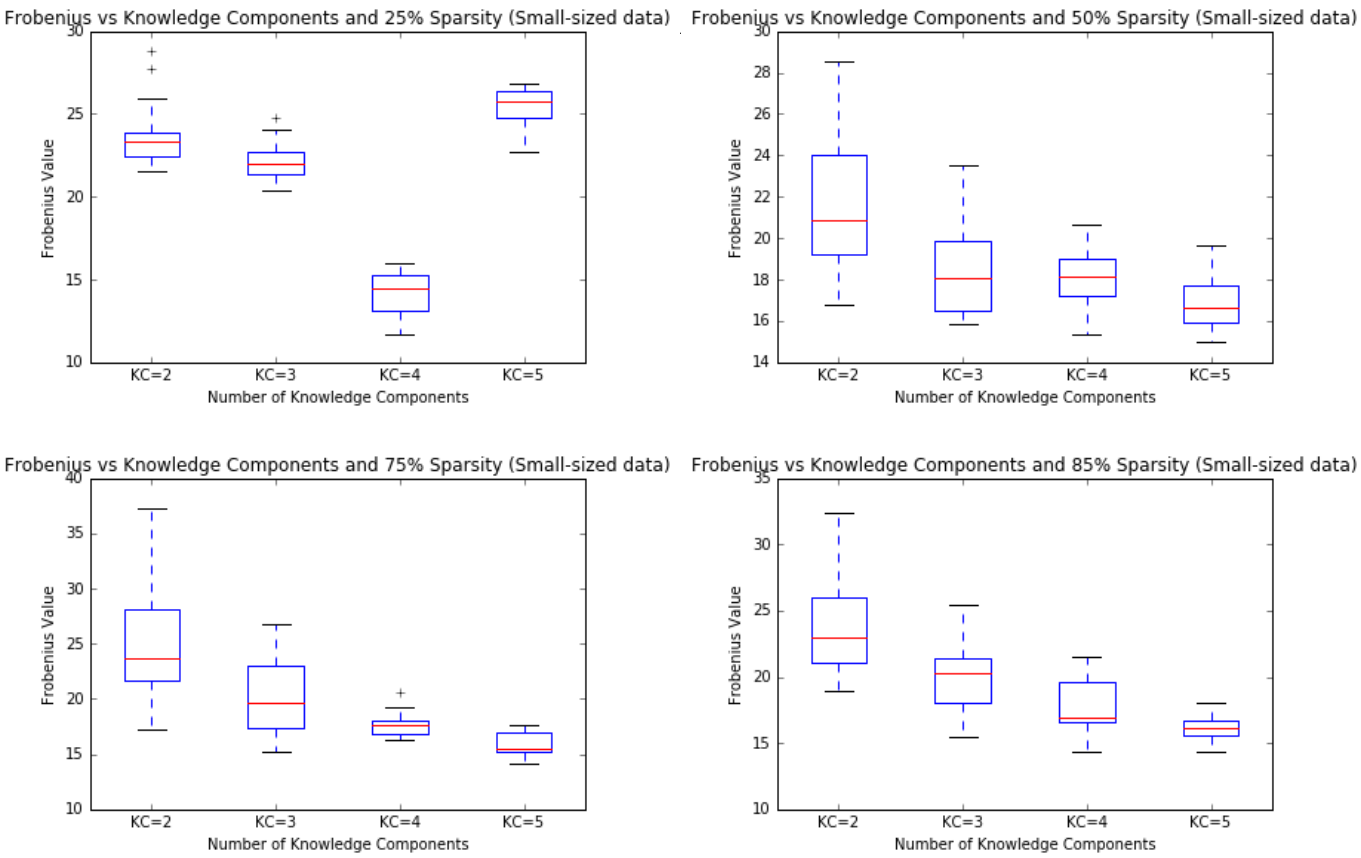

Figure 7.2: Frobenius Norm with different number of knowledge components and sparsity rates in small-sized instances.

where $p_{ijt} = P(\mathcal{Y}_{ijt} = 1)$ is the true probability determined by the second synthetic data model and $\widehat{p}_{ijt} = P(\mathcal{Y}_{ijt} = 1|\mathcal{X}_{:jt}, \theta_j, d_i)$ is estimated by LBS model.

7.6.2 Computational Results of Predictive Model

This section explores the computational results of LBS model on three different datasets. The LBS model performance to predict the probabilities that students will answer the given questions correctly is evaluated using small-, medium- and large-sized instances. As described in the previous section, several measures are employed to monitor the model output. The most important indexes reflecting accuracy of prediction are Frobenius Norm and Mean Square Error of the test set. The test set is randomly chosen and sparsity rate shows that the percentage of unobserved data. Table reports the computational results.

In small-sized instances, AIC, AIC_c and BIC select the model with a smaller number of knowledge components while with 25% sparsity, ($K = 3$) provides the smallest MSE and Frobenius Norm. When the sparsity increases, models with higher value of K perform better. Figure 7.2 depicts Frobenius Norm with $K = 2, 3, 4$ and 5 and sparsity rates 25%, 50%, 75% and 85%. As the number of knowledge components (K) increases, the Frobenius Norm obtained by (7.45). It should be noted that the number of knowledge components in the synthetic dataset is $K = 3$. The similar pattern is observed in large-sized instances (see Figure 7.3).

Sparsity of observations is the key challenge in student modeling due to the fact that

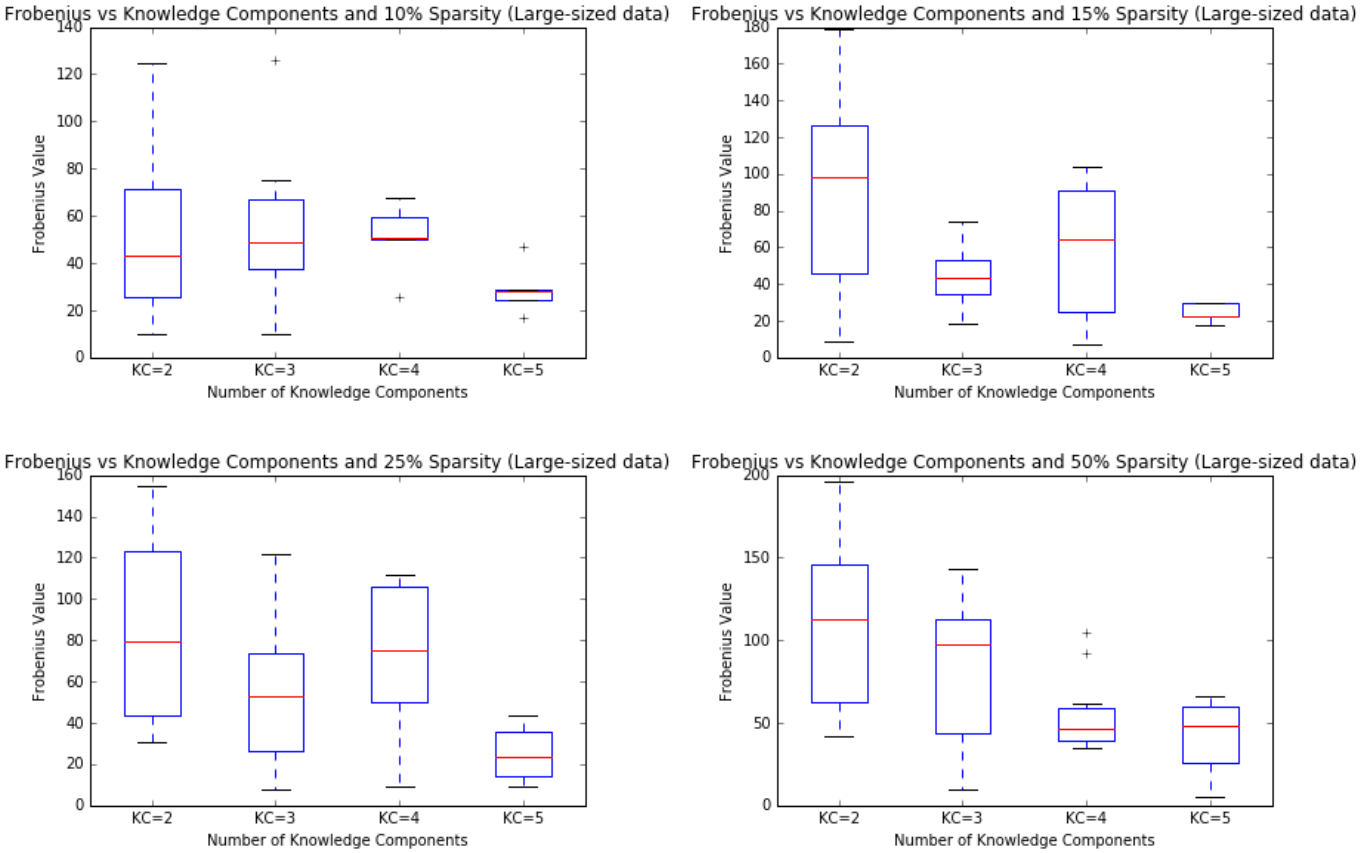

Figure 7.3: Frobenius Norm with different number of knowledge components and sparsity rates in large-sized instances.

Table 7.2: Simulation Results of LBS model using with different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE
Small-sized	25%	2	1194.542	1228.116	4447.313	27.025	-157.271	23.759	0.047
		3	1830.874	1907.822	6710.031	129.61	-255.437	22.156	0.041
		4	2152.618	2292.069	8658.16	488.18	-196.309	14.211	0.017
		5	2831.022	3053.263	10962.95	540.118	-315.511	25.482	0.054
	50%	2	1306.092	1339.666	4558.864	10.818	-213.046	21.886	0.041
		3	1714.306	1791.255	6593.463	56.058	-197.153	18.575	0.029
		4	2061.256	2200.707	8566.798	196.021	-150.628	18.223	0.028
		5	2440.44	2662.681	10572.369	308.321	-120.22	16.805	0.024
	75%	2	1029.383	1062.957	4282.155	7.328	-74.692	24.548	0.052
		3	1407.218	1484.167	6286.375	12.744	-43.609	20.241	0.035
		4	1923.5	2062.951	8429.042	155.999	-81.75	17.676	0.026
		5	2285.38	2507.621	10417.308	246.231	-42.69	15.898	0.021
	85%	2	992.348	1025.922	4245.119	4.977	-56.174	24.233	0.05
		3	1437.91	1514.858	6317.067	11.238	-58.955	19.824	0.033
		4	1858.569	1998.02	8364.111	134.084	-49.284	17.714	0.027
		5	2170.142	2392.383	10302.07	305.319	14.929	16.183	0.022
Medium-sized	15%	2	4167.096	4203.676	15314.835	230.166	-883.548	48.669	0.081
		3	5356.921	5439.833	22078.528	1007.646	-878.461	38.02	0.058
		4	6796.212	6944.73	29091.689	1577.44	-998.106	39.676	0.06
		5	8305.001	8538.848	36174.347	3165.291	-1152.5	40.087	0.07
	75%	2	2970.001	3006.58	14117.74	111.887	-285.001	57.248	0.101
		5	6519.715	6753.563	34389.061	1220.986	-259.858	40.857	0.091
Large-sized	10%	2	68808.828	68851.734	114920.142	2135.583	-30404.414	90.264	0.17
		3	72169.621	72266.411	141336.592	4967.316	-30084.81	45.135	0.13
		4	73792.609	73965.138	166015.237	5950.838	-28896.305	58.294	0.14
		5	76934.907	77205.205	192213.192	17161.476	-28467.454	24.131	0.091
	15%	2	70999.532	71042.438	117110.846	3529.391	-31499.766	51.154	0.141
		3	74907.271	75004.061	144074.242	4934.701	-31453.636	53.286	0.145
		4	78422.986	78595.514	170645.614	5593.479	-31211.493	50.746	0.138
		5	81397.894	81668.192	196676.179	11225.984	-30698.947	29.032	0.097
	25%	2	63581.644	63624.55	109692.958	2171.323	-27790.822	84.103	0.183
		3	64583.533	64680.323	133750.504	5019.169	-26291.766	54.898	0.146
		4	67540.763	67713.292	159763.391	4292.245	-25770.382	70.313	0.179
		5	71785.313	72055.611	187063.598	13921.097	-25892.657	25.252	0.091
	50%	2	44837.847	44880.753	90949.161	1197.413	-18418.923	109.304	0.219
		3	46681.465	46778.256	115848.436	2227.537	-17340.733	80.809	0.182
		4	50107.891	50280.419	142330.519	4615.723	-17053.945	55.2	0.145
		5	53976.412	54246.71	169254.697	7968.293	-16988.206	41.177	0.124

students answer only few questions in ITSs with large number of questions. Therefore, the lack of enough observation makes inferring about student conceptual learning very difficult. However, matrix and tensor factorization methods perform well with sparse data. Figure 7.4 illustrates the impact of sparsity on Frobenius norm with two and five knowledge components. The more sparse data, the higher value of Frobenius norm is obtained that shows a larger error in prediction.

7.6.3 Computational Results of CMPC

This section explores the performance of CMPC to find an optimal sequence of questions that each student should be given. The CMPC personalizes learning experience by taking

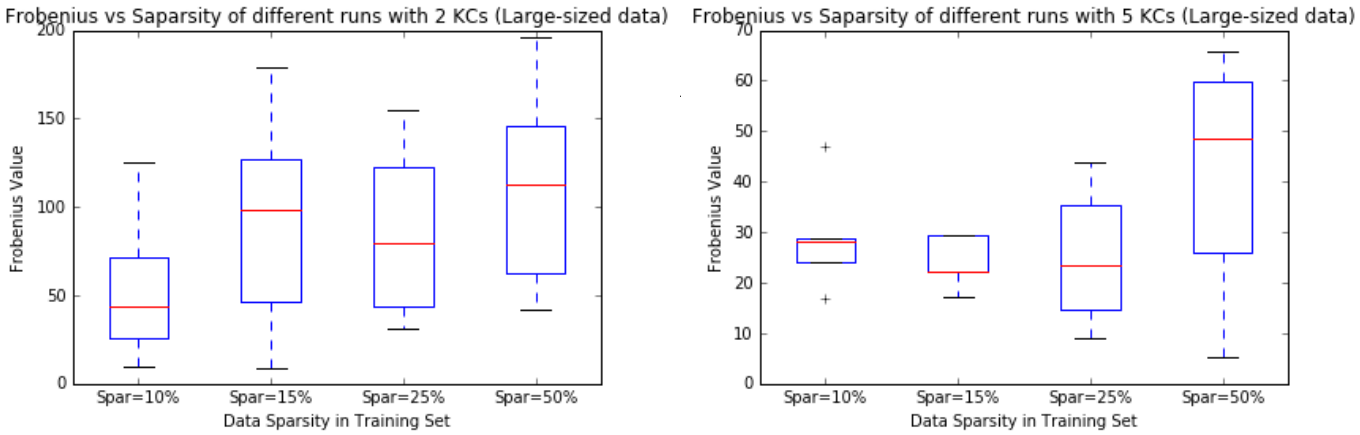

Figure 7.4: Frobenius Norm with different sparsity rates for 2 and 5 knowledge components in large-sized instances.

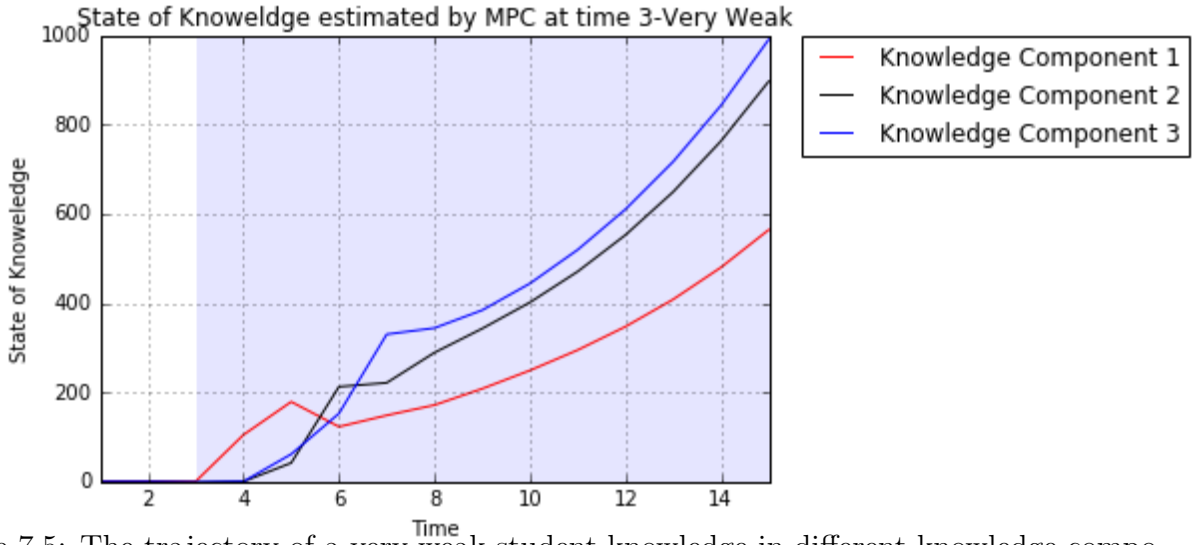

Figure 7.5: The trajectory of a very weak student knowledge in different knowledge components in the course after $T = 3$.

into account the parameters from LBS model that are specified to each student. In order to evaluate effectiveness of the model, a course with five students is assumed. These students ranging from very weak to extraordinary, need to answer at least one question in each time stamp. Originally, 100 questions are available in the question bank and students have only one chance to try a specific question. After that, the question is no longer available to the student.

Figure 7.5 shows trajectories of knowledge components based on the optimal sequence of questions given to a very weak student in the class. According to a simulator that mimics students dynamics, if the optimal sequence of questions is given to the student, he is able to improve his knowledge state after four or five steps. Figure 7.6 depicts a weak student learning in three knowledge components after optimizing the sequence of questions. The student has still some troubles but performs better than the previous case. Figure 7.7 shows how an extraordinary student makes progress by working on the optimal sequence of questions.

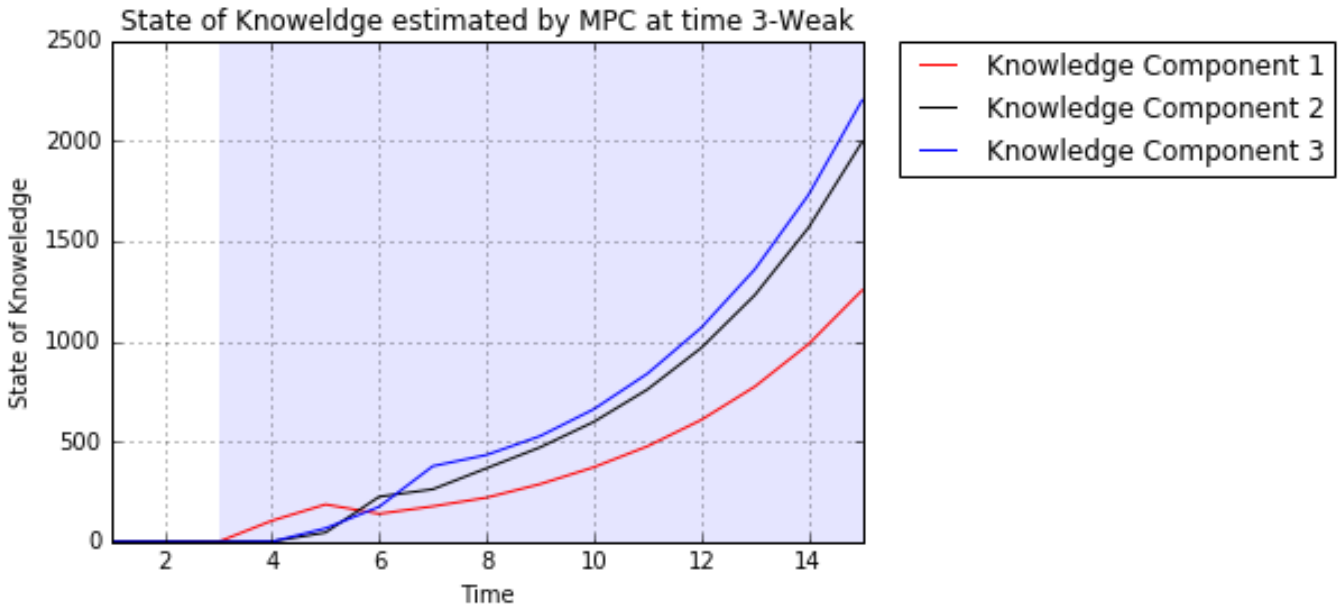

Figure 7.6: The trajectory of a weak student knowledge in different knowledge components in the course after $T = 3$.

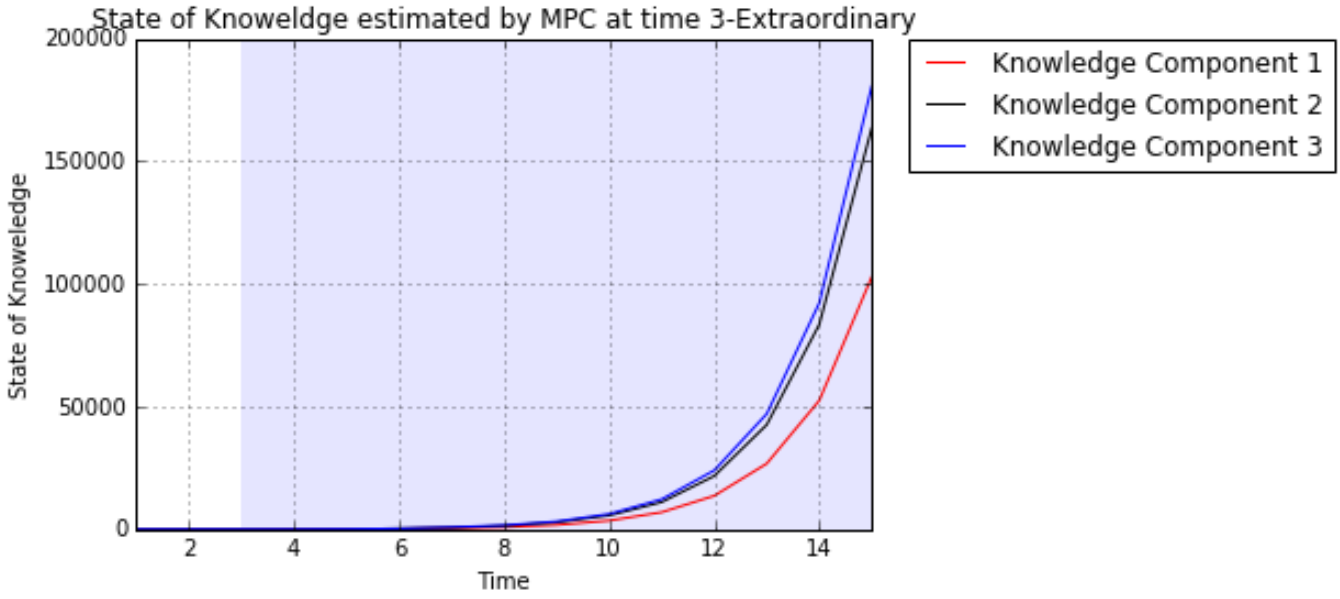

Figure 7.7: The trajectory of an extraordinary student knowledge in different knowledge components in the course after $T = 3$.

7.7 Conclusion

This chapter offers a new framework in student modeling that not only specifically consider the impact of each question answered by the student on his learning experience but it distinguishes also between the learning gains obtained if the student correctly or incorrectly answers the questions. The proposed framework predicts students performance and evaluate their conceptual learning at each moment of time. The second component of this framework uses the predictions and student state of knowledge and optimize the sequence of questions that students should be given in a personalized manner.

The proposed model called Learning By Solving student model (LBS) is based on a constrained tensor factorization model that captures the interconnection between different knowledge components. It is assumed that student state in each knowledge components representing the conceptual learning, is a continuous random variable that follows a Gaussian distribution. The mean of this distribution is assumed to be a function of previous state and the given question. LBS is a comprehensive model that relaxes restrictive assumptions made by other latent factor models and specifically pays attention to the effect of questions on conceptual learning.

LBS model is employed as a predictive model to optimize teaching policy by finding the optimal sequence of questions that each student is assigned to in a personalized way. To find personalized optimal teaching policies, the sequential decision making is formulated as a Reinforcement Learning (RL) problem. However, the resulted RL problem is computationally intractable due to the fact that each question can be used only once; therefore, a Constrained Model Predictive Control (CMPC) approach is selected instead of RL. The model predictive control offers an adaptive mechanism to estimate students learning dynamics based LBS and optimize the sequence of questions individually for each student. The simulation results illustrate the LBS model performs well in predicting students performance as well as in inferring students conceptual learning. Running the CMPC reveals that the model distinguishes between students capabilities in finding the optimal teaching policy.

Chapter 8

Summary and Future Research

This dissertation offers the socio-technical pedagogical paradigm "crowdlearning" that creates a collaborative problem-posing and problem-solving environment for STEM students in physical and virtual classrooms. This study also addresses obstacles in development of Intelligent Tutoring Systems (ITS) including scalability, modeling of student conceptual learning, personalizing optimal teaching policy by incorporating advances in machine learning and operations research techniques. The main idea behind this research is to explain how students learn by observing their performance and how instructors can personalize learning contents to boost their knowledge level in course materials. To end this, a framework for formulating and predicting dynamics of student learning based on new tensor factorization models is introduced and aligned with a model predictive control approach that personalizes the optimal teaching policy for each student.

8.1 Conclusion

Modeling student conceptual understanding is an integral component in Intelligent Tutoring Systems, wherein students are given fast feedback through an online platform. A appropriate model not only accurately predicts student performance but it also help instructors to infer student conceptual learning. The benefit of using such reliable systems in educational domain is to reduce pressure on teaching and learning as well as saving time and effort for both instructors and students.

8.1.1 Crowdlearning online platform

The first part of this study mainly focuses on developing a web-based tool that has been hosted in "crowdlearning.eng.buffalo.edu" to engage STEM students and instructors in problem posing and solving activities, wherein the students experience deeper learning by gaining new perspectives on the use of subject-matter concepts and by learning with and from each other. The core-stone idea of Crowdlearning is that of a self-sustaining problem-posing and problem-solving environment, where the students play the roles as (A) the creators of subject-focused problems (problem statements/formulations with answer alternatives, hints, correct answers with explanations, etc.) and (B) problem solvers all on the online platform.

8.1.2 New Synthetic Data Generation Models

The research extends two synthetic data generating models that enable one exploring students conceptual learning in more depth. The presented models take advantage of Bayesian Knowledge Tracing and Latent Factor Analysis, the state-of-the-art models in modeling student learning and predicting their performance. The first model considers student learning as a discrete random variable and employs the idea of Factorial Hidden Markov Model to build the transition probability matrix and generate observations. The second model postulates that student learning dynamics is a continuous quantity captured by a first order difference equation. The student learning models developed throughout this dissertation are tested with the proposed synthetic data.

8.1.3 Parallel Sparse Factor Analysis

This study improves current ITS state-of-the-art student learning model in terms of computational performance and proposes a Parallel Coordinate Descent algorithm in shared memory systems that significantly makes the recently proposed SPARse Factor Analysis (SPARFA) scalable. This new implementation enables student modeling analyses at scale, by presenting a parallel algorithm for performing SPARse Factor Analysis (SPARFA). SPARFA is accepted as a state-of-the-art machine learning approach for estimating student knowledge levels and predicting their performance in not-yet-taken educational tasks. However, with the Likelihood Maximization (LM) formulation behind it resulting in a non-convex optimization problem with many local minima, the scalability has been a challenge for SPARFA. While the original method employs an alternating optimization approach to approximately solve the SPARFA LM problem, this work employs a Parallel Coordinate Descent (PCD) algorithm parallelized in the shared memory setting. The performance of PCD is evaluated on varied problem instances with synthetic data and is found to successfully solve instance with up to 7.5 Million variables, which makes it suitable to make inference from the data of any real-world Massive Open Online Course.

8.1.4 New Predictive Models of Student Learning

This work also improves Latent Factor Analysis models and offers new methods for dealing with student performance prediction, framed as a Probabilistic Matrix Factorization problem. The proposed framework presents new constrained tensor factorization models enabling one to extract meaningful patterns from highly sparse data. The key challenge of modeling and predicting students performance lies in the inference of their conceptual understanding while recognizing both the temporal dynamics of knowledge acquisition and the differences in individual learning abilities. Responding to this challenge, new constrained tensor factorization models in probabilistic formulation are proposed. Inferring student conceptual learning results in a Likelihood Maximization formulation that encompasses a non-convex optimization problem. The well-knowm Alternating Direction Method of Multipliers (ADMM) technique is efficiently implemented to tackle with the constrained non-convex optimization problems. The models feature well-interpretable parameters, and are shown to compare favorably against two competing tensor factorization methods.

8.1.5 Personalize Optimal Teaching Policy

Finally, this dissertation expands a new framework in student modeling that not only takes into account the effect of each question on student conceptual learning but it distinguishes also between the gains obtained form correctly and incorrectly answered questions. The proposed framework that is founded on top of a constrained tensor factorization models the interconnection between different knowledge components as well. This framework called Learning-By-Solving (LBS), is utilized as a predictive model to optimize teaching policy by finding the optimal sequence of questions that each student is assigned to, in a personalized way. To find personalized optimal teaching policies, the corresponding sequential decision making is formulated as a Reinforcement Learning (RL) problem. The resulted RL problem is computationally intractable due to the fact that each question can be used only once; therefore, a Constrained Model Predictive Control (CMPC) approach is exploited as an alternative solution to RL. The model predictive control offers an adaptive mechanism to estimate students learning dynamics based on LBS model and optimize the sequence of questions personalized for each student. The simulation results illustrate LBS can predictive students performance very well and the CMPC optimizes teaching policies for students and personalizes the learning experience.

8.2 Future Directions

This dissertation opens new areas of student modeling and personalizing learning contents in ITS. As mentioned before, the use of ITSs has witnessed a significant jump in educational institutes with advent of MOOCs. Student modeling is still a new line of research and there are still a lot of unanswered questions. Although several methods have been developed to address some of the challenges and improve the current state-of-the-are models, a number of lines are still open for future work that are outlined as follows.

8.2.1 Future of Crowdlearning

Connecting the crowdlearning problem bank use with educational materials (textbook excerpts, videos, etc.) can produce comprehensive subject matter knowledge bases and enable the research of individually tailored learning roadmaps, enhancing the learning experiences of those engaged and not engaged in any formal training. In particular, crowdlearning can be used with e-learning tools to enhance the appeal of this form of educational practice. Its dissemination is likely to snowball, as new adopters will be benefitting from all the products (problems bank refinements) contributed by prior adopters; moreover, the problems most helpful to learners will be automatically recognized as such. Crowdlearning opens possibilities for deeper learning in any field where multiple-choice questions can be used for assessment, not limited to the university setting or STEM (e.g., it can be used for SAT preparation). The crowdlearning idea can be expanded to the crowdsourced creation of study "cases" often used in business, law, etc. Thus, Crowdlearning bridges the gap between crowdsourcing and online learning as envisioned by Williams et al. (2015).

The next versions of online learning platforms are primed to benefit from the models introduced in this dissertation and crowdlearning-specific algorithms, e.g., for enabling and

improving organized problem assessment. We also envision growing a community of teachers and researchers interested in adopting and developing crowdlearning. Such individuals can serve on voted-in Editorial Boards of instructors who will oversee the formation of publically accessible problem banks.

8.2.2 Parallel Implementation of Proposed Models

Utilizing student learning models at scale where thousands of students allover the world register in a course is not achievable unless the computational resources are used in smarter way. Parallel computing is a new field of research so that there exist a number of opportunities to develop parallel machine learning models. In this study, only one possible approach out of many was touched and there are still many other algorithms that once can develop to improve the performance. For instance, a distributed systems approach can be used instead of the shared memory systems for larger problems for non-convex optimization. In addition, Block Coordinate Descent is not the only optimization algorithm that can be parallel-ized.

In this study, several tensor factorization-based models have been proposed. Tensor Factorization models are criticized to increase the number of parameters and making the optimization task very difficult, however, using the parallel approach can alleviate this issue. In particular, the constrained model predictive control component offer a series of optimization problem that are absolutely independent so that it does not make sense to run in in sequential specially for large class settings.

8.2.3 Bayesian Versions of Student Models

Several student models have been developed in this research based on tensor factor techniques where one needs to infer many parameters such as $\mathbf{C}, \mathbf{W}, \mathbf{V}, \mathcal{X}$ and so on. Using a Bayesian approach based on Markov chain Monte Carlo (MCMC) sampling method can improve the student models in terms of dealing with uncertainties. Although point estimation method of the parameters of interest are computationally less expensive, a fully Bayesian approach to the proposed student models enables one to compute full posterior distributions for $\mathbf{C}, \mathbf{W}, \mathbf{V}, \mathcal{X}$ and compensates the computational complexity with several considerable benefits in the context of learning and content analytics (Lan et al., 2014).

In addition, considering full posterior distributions enables the computation of informative quantities such as credible intervals and posterior modes for all parameters. Moreover, knowing the fact that MCMC methods search the full posterior space, they can escape from local minima. Finally, the hyperparameters used in Bayesian approach generally ease interpretation of model parameters.

8.2.4 Heuristics for Reinforcement Learning Formulation

As discussed before, the main challenge of employing reinforcement learning model is that each action (i.e., question) can be effectively used only once in this application. In other words, if a student sees a particular question, this question cannot significantly boost the student learning gain in the second time. Even if one assumes that the student may forget

the question by incorporating a forgetting factor, this assumption does not help to make the problem tractable.

The consequence of considering one-time used questions results in a combinatorial optimization problem that makes the reinforcement learning approach ineffective in real time applications. Developing a good heuristic to deal with this issue may justify to make use of RL approach.

8.2.5 POMDP Approach to Optimal Teaching Policy

POMDP is a potential solution for finding optimal teaching policy because the observations stored from students performance are noisy and incomplete. In the literature review, some models that incorporates lucky guesses, careless mistakes and forget factor were reviewed. The structure of this problem is a good fit for POMDP although there are several challenges in adapting POMDP approach in this case. In appendix D, the idea of using POMDP and proposed algorithm is discussed.

Appendix A

Supplementary to Chapter 2

A.1 Crowdlearning: More Snapshots

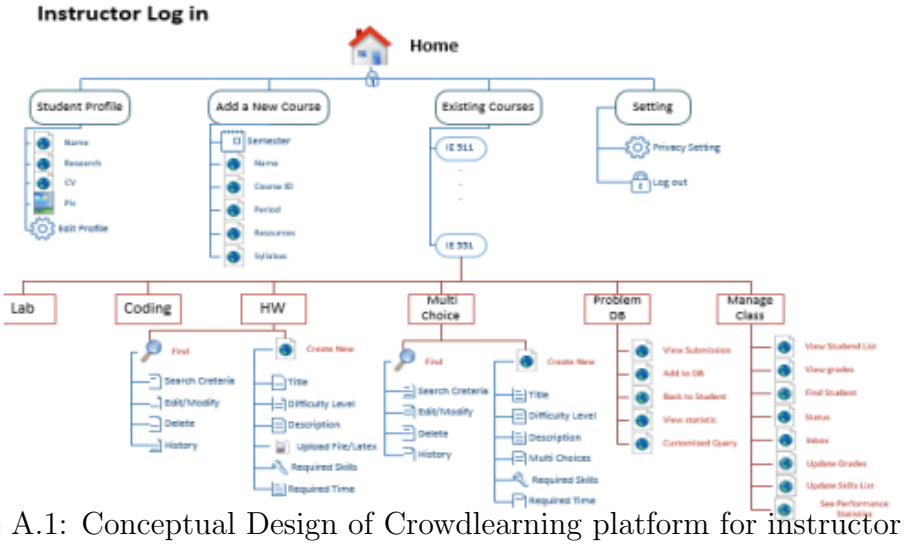

Figure A.1: Conceptual Design of Crowdlearning platform for instructor log-in.

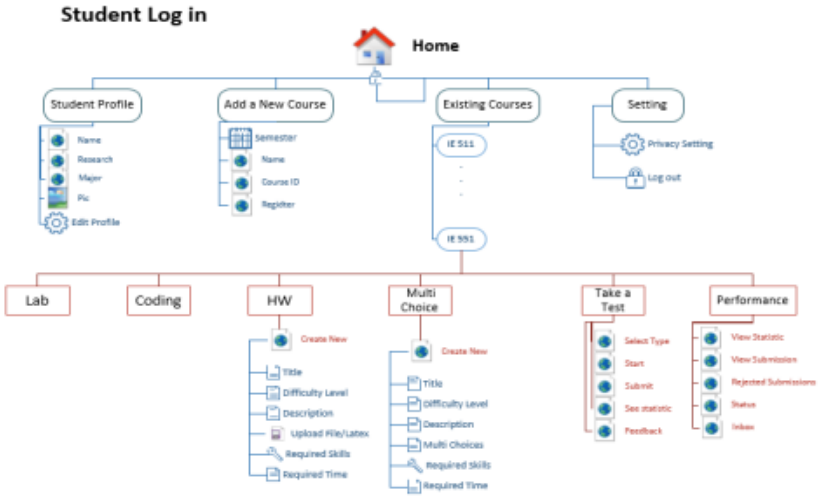

Figure A.2: Conceptual Design of Crowdlearning platform for student log-in.

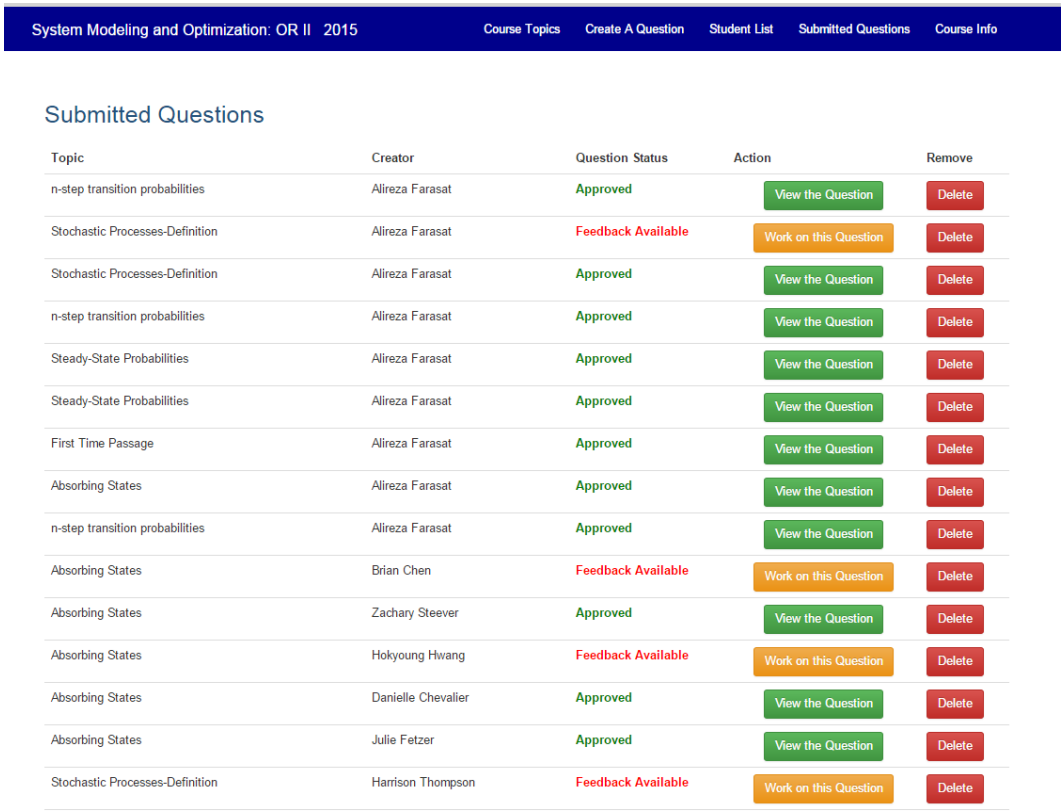

Figure A.3: The instructor has the list of all problems posed by students and can check the status of each question. Crowdlearning enables one to not only edit the problem by herself but also give a useful feedback to the creator. In the next step, students will be also considered as evaluators and they will have a customized page that help them with the evaluation.

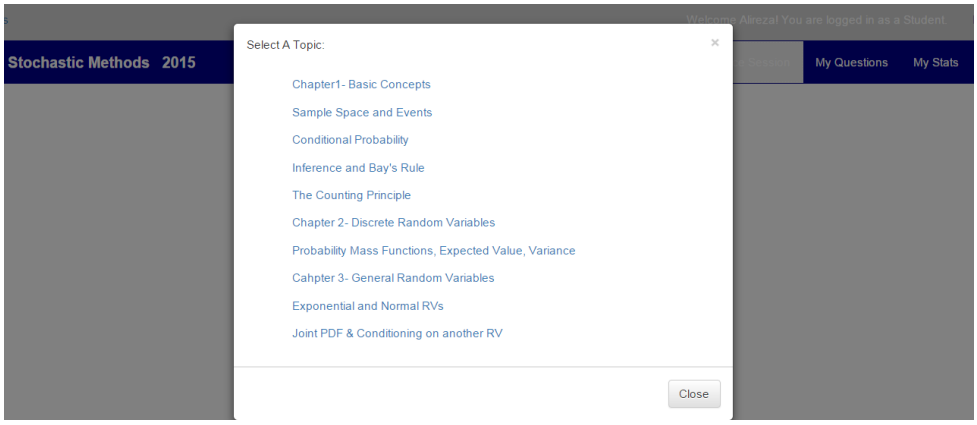

Figure A.4: CrowdLearning has a Practice Session tool that offers the students to select a topic from all topics taught in the course. A question from several questions available in the course is selected and proposed to the student. In the next version of CrowdLearning, a self-optimizing tool is developed to more systematically customize the sequence of the questions that each student should answer. To accomplish this an intelligent agent is developed to track the performance history of each students and infer about her state of knowledge. .

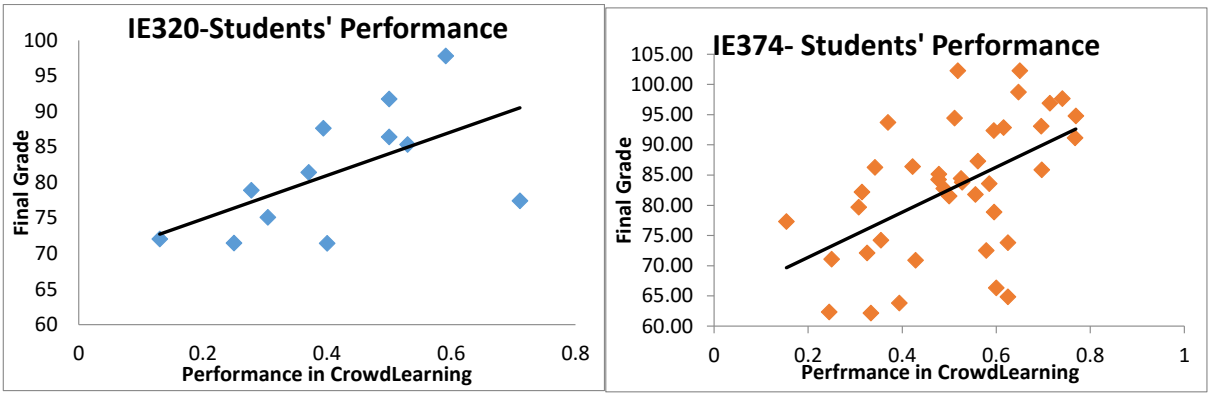

Figure A.5: This figure shows the number of students and number of questions they proposed in the courses that crowdleaning platform was used.

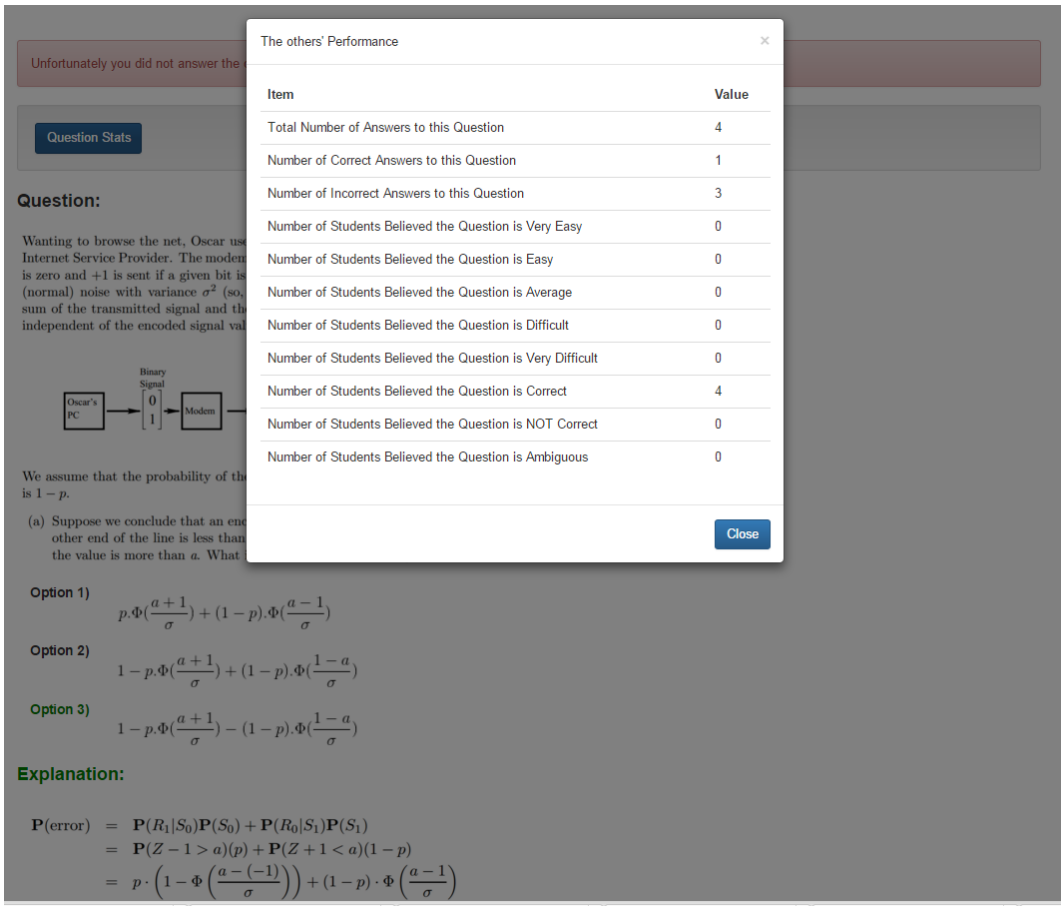

Figure A.6: Once the question is submitted, the correct answer along with and a concise explanation is provided so the students can figure out what the correct solution is or what she has done incorrectly. In addition, statistics on this question is available so the student is informed what other peers have done on this question. Next version of CrowdLearning will uses some packages to more visually illustrate statistics. In addition, it is required that students give feedback about how much helpful the solution was or whether the solution is

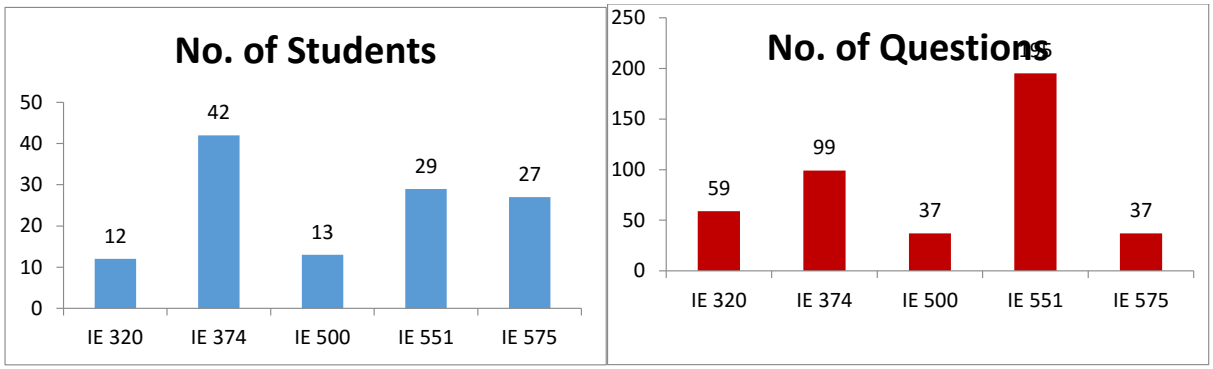

Figure A.7: This figure shows the number of students and number of questions they proposed in the courses that crowdleaning platform was used.

Appendix B

Supplementary to Chapter 4

B.1 Factorial Hidden Markov Model

Factorial Hidden Markov Model is a generalized HMM where a collection of discrete state variables are taken into account Ghahramani (2001). in this problem, state variables indicate the level of mastery in each knowledge component. Let $X_t^k \in \{0, 1, 2, 3\}\ \forall\ t = 1, \ldots, T\ k = 1, \ldots K$ denotes the status of knowledge component k at time t where 0, 1, 2 and 3 indicate *not knowing*, *weak*, *shaky* and *solid* knowledge, respectively. Furthermore, T_n is a parameter associated with student n's inherent capabilities $n = 1, \ldots, N$ and D_q denotes inherent difficulty of question q assuming that there exist N student and Q questions. Let $Y_t^q \in \{0, 1\}$ represent the student n's performance on question q which 1(0) means that the student has answered the question correctly(incorrectly). Figure 1 illustrate a FHMM for identifying students' knowledge status.

The FHMM depicted in Figure B.1 assumes that all knowledge components are related to a

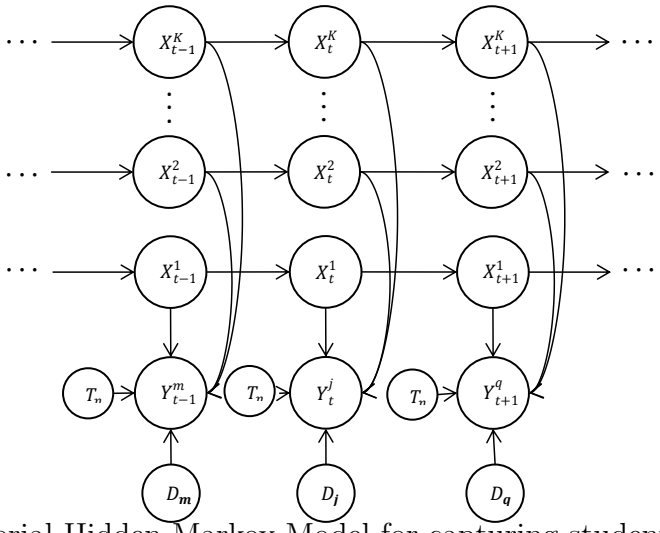

Figure B.1: Factorial Hidden Markov Model for capturing student u's knowledge status.

particular question which is not a true assumption (dependencies between X_t^k and Y_t^q. This assumption is not realistic because it is not possible that all questions cover all knowledge

components. Such issue necessitates structural learning of FHMM knowing that main part of the structure (dependency between X_{t-1}^k and $X_t^k \; \forall \; k$ is known).

B.2 FHMM Parameters Estimation

Estimating FHMM parameters is the main challenge in this research since X_t^k follows unknown discrete distribution and there exist a limited number of observation. Using dependencies relations between variables in the FHMM, the joint probability of the sequence of latent states and observation can be captured as follows:

$$P(\boldsymbol{X_t}, Y_t^q) = \prod_{t=1}^{T} P(\boldsymbol{X_t}|\boldsymbol{X_{t-1}}) P(Y_t^q|\boldsymbol{X_t}, Z_n, D_q),$$

where $\boldsymbol{X_t} = \{X_t^1, \ldots, X_t^K\}$ and

$$P(\boldsymbol{X_t}|\boldsymbol{X_{t-1}}) = \prod_{k=1}^{K} P(X_t^k|X_{t-1}^k),$$

therefore, if all questions for student n is considered the likelihood function has the following form:

$$\max \prod_{q=1}^{Q} \prod_{t=1}^{T} P(\boldsymbol{X_t}|\boldsymbol{X_{t-1}}) P(Y_t^q|\boldsymbol{X_t}, Z_n, D_q). \tag{B.1}$$

$$(P2): \quad \max_{\mathbf{W},\mathbf{C},\mathbf{V}} \quad \sum_{(i,j,t)\in\Omega_{obs}} log(p(Y_{ij}^t|\mathbf{w}_i, \mathbf{c}_j, \mathbf{v}_t))$$

$$s.t.$$

$$\|\mathbf{w}_i\|_0 \leq \delta \; \forall \; i \tag{B.2}$$

$$\|\mathbf{w}_i\|_2 \leq \beta \; \forall \; i \tag{B.3}$$

$$\|\mathbf{C}\|_F = \xi \tag{B.4}$$

$$\|\mathbf{V}\|_F = \delta \tag{B.5}$$

$$\sum_{k=1}^{K} c_{ki}(v_{kp} - v_{kt}) \leq \epsilon \;\; \forall \; i \;\; p < t \;\; p, t = 1, \ldots, T \tag{B.6}$$

$$W_{ik} \geq 0 \; \forall \; i, k \tag{B.7}$$

$$p(Y_{ij}^t|\mathbf{w}_i, \mathbf{c}_j, \mathbf{v}_t) = \Phi(< \mathbf{w}_i, \mathbf{c}_j, \mathbf{v}_t >)^{Y_{ij}^t} [1 - \Phi(< \mathbf{w}_i, \mathbf{c}_j, \mathbf{v}_t >)]^{1-Y_{ij}^t}$$

Appendix C

Supplementary to Chapter 6

C.1 Tensor Definitions

- **Norms of Matrix.** Some norms of matrix such as ℓ-0, ℓ-1, ℓ-2 and Frobenius norms are used in this paper. ℓ-0 norm, $\|\mathbf{A}\|_0$ indicates the number of non-zero elements in matrix $\mathbf{A}$. Norms ℓ-1, and ℓ-2 which is the square root of the largest eigenvalue of the positive-semidefinite matrix A^*A are defined as follows:

$$\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{n} |A_{ij}|$$
$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{max} \mathbf{A}^\star \mathbf{A}},$$

  where λ_max is the largest eigenvalue of the positive-semidefinite matrix $\mathbf{A}^\star\mathbf{A}$ and $\mathbf{A}^\star$ denotes the conjugate transpose of $\mathbf{A}$. Frobenius norm is defined as follows;

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} |A_{ij}|^2 \right)^{\frac{1}{2}}.$$

  It is easy to show that $\|\mathbf{A}\|_2 \leq |\mathbf{A}\|_F$.

- **Tensor Fibers.** A fiber, a generalization of matrix rows and columns to a higher order, indicates a sequence of elements along a fixed mode when only one index is not fixed and others are fixed. Therefore, a mode-1 fiber of a tensor represents a matrix column, a mode-2 fiber of a tensor shows a matrix row. A mode-3 fiber in a tensor is a tube.

- **Tensor Slices.** The results of fixing all but two indexes in a tensor is a two-dimensional array, i.e. matrix which is called slice. For example, $\mathcal{Y}_{i::}$ is a 2-D matrix where index i is fixed. Similarly, $\mathcal{Y}_{ij:}$ denotes a vector wherein indexes i, j of $\mathcal{Y}$ are fixed.

C.2 Performance of Models

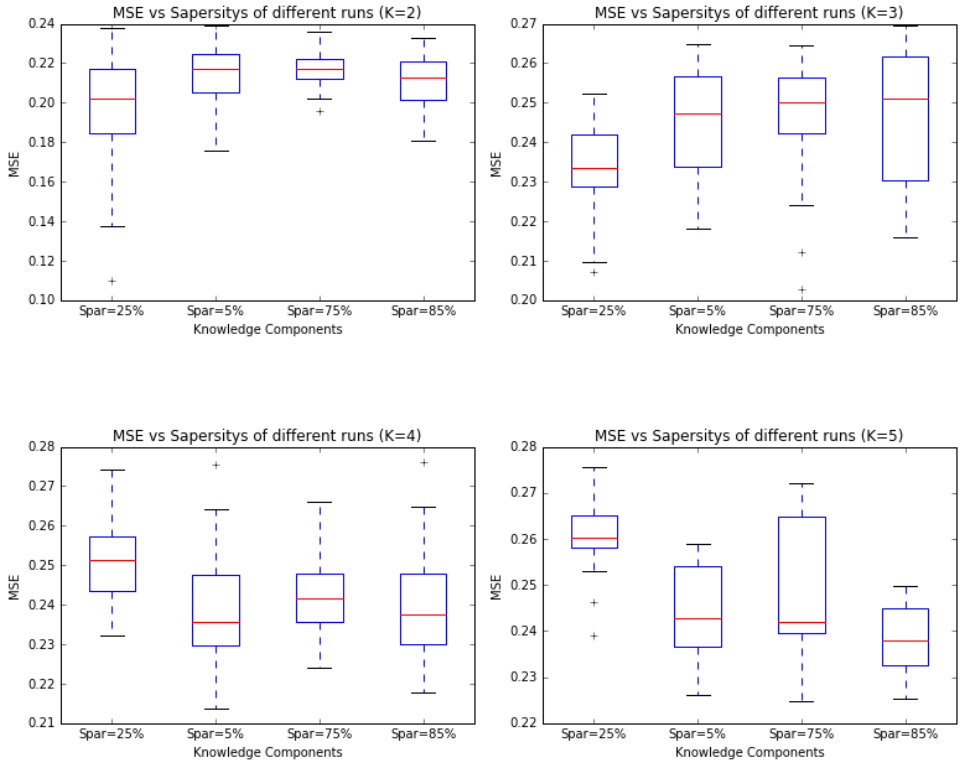

Figure C.1: Comparison between MSE vs observation sparsity in model 1 (SSM) with the small-sized problem (different number of Knowledge Component) .

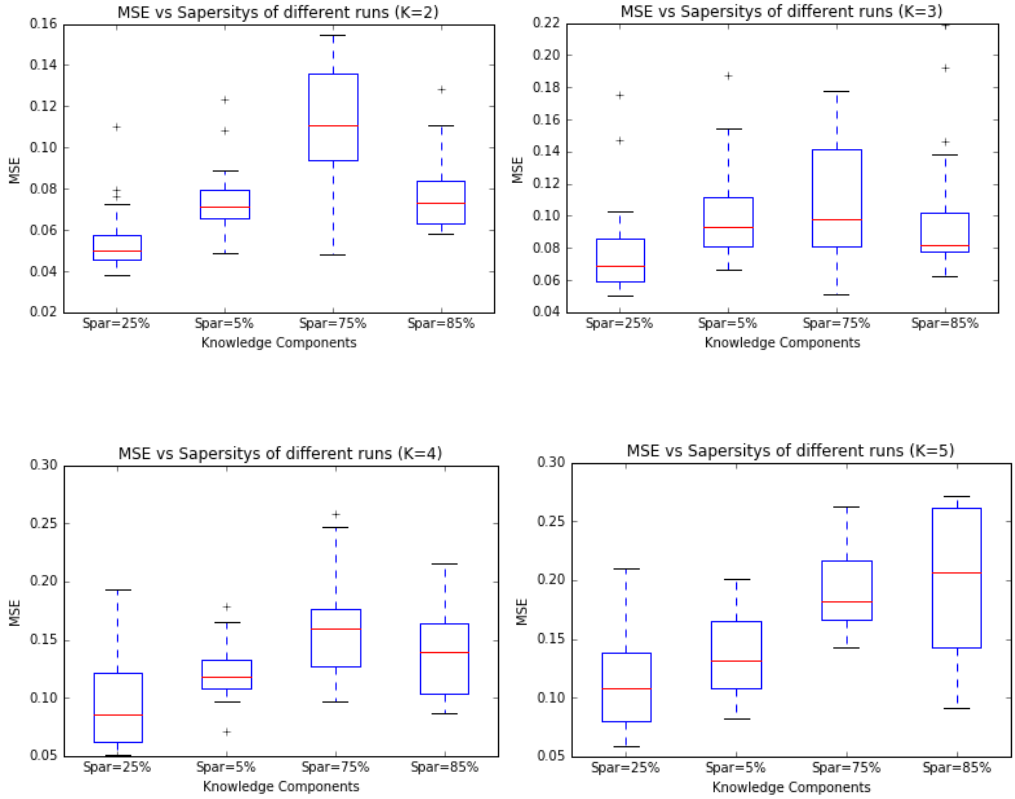

Figure C.2: Comparison between MSE vs observation sparsity in model 2 (HSM) with the small-sized problem (different number of Knowledge Component) .

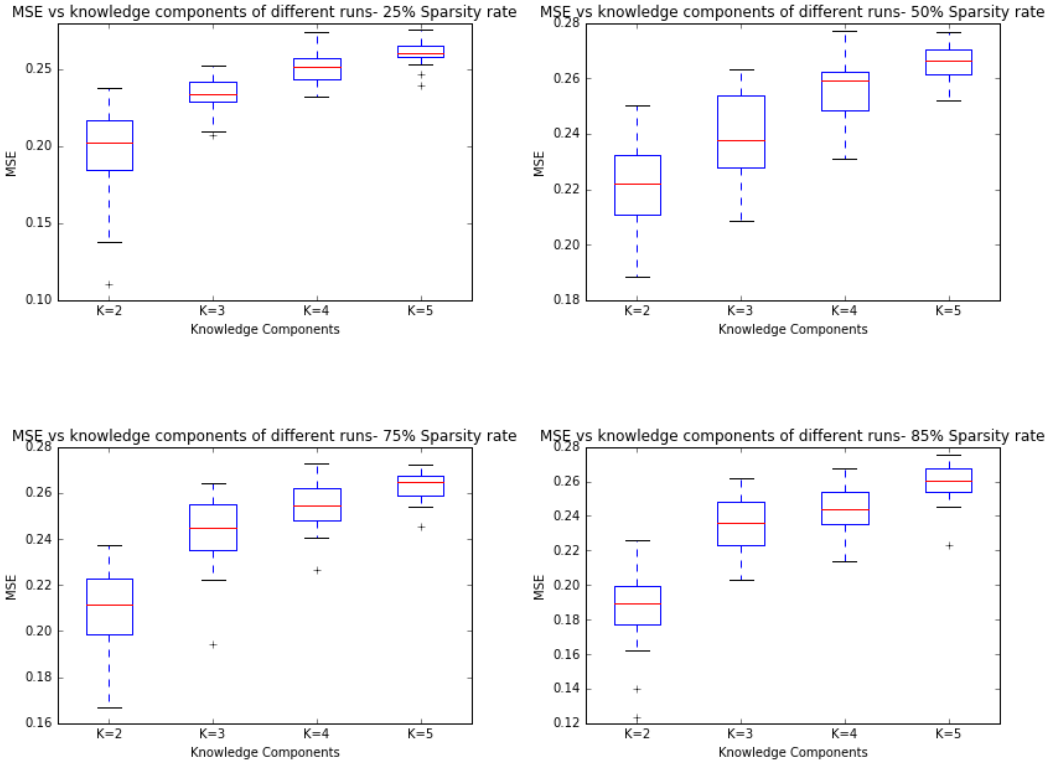

Figure C.3: Comparison between MSE vs Knowledge Components in model 1 (STF) with the small-sized problem (different sparsity rate) .

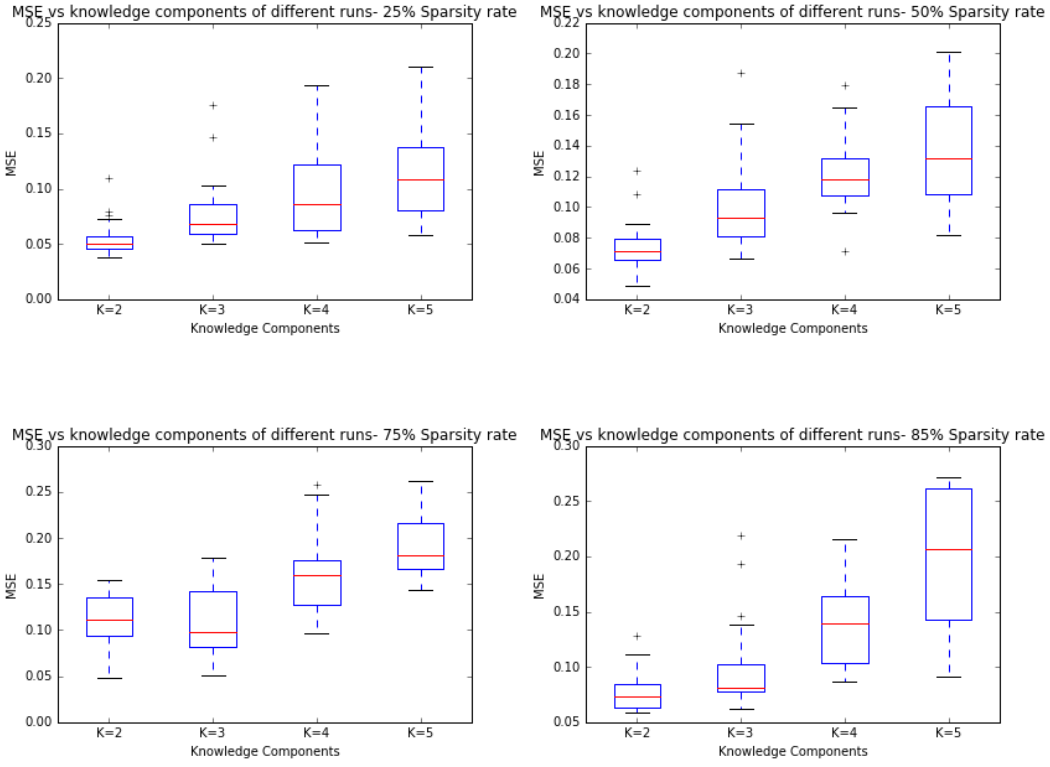

Figure C.4: Comparison between MSE vs Knowledge Components in model 2 (CTF) with the small-sized problem (different sparsity rate) .

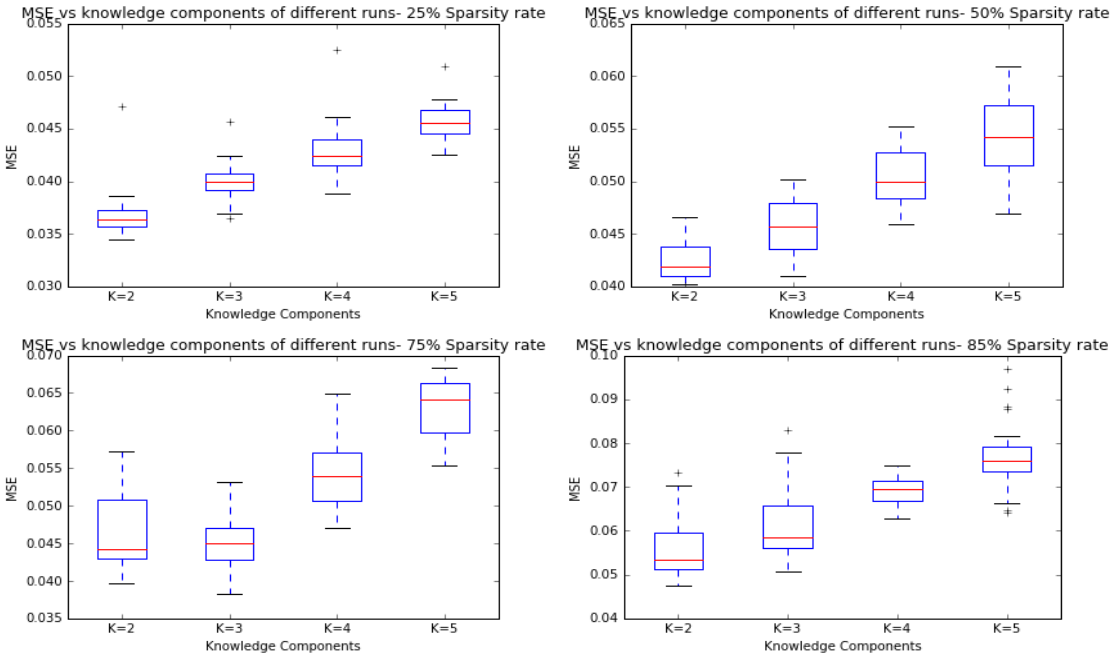

Figure C.5: Comparison between MSE vs Knowledge Components in model 3 (PTF) with the small-sized problem (different sparsity rate) .

Appendix D

Supplementary to Chapter 7

D.1 Model Predictive Control Plots

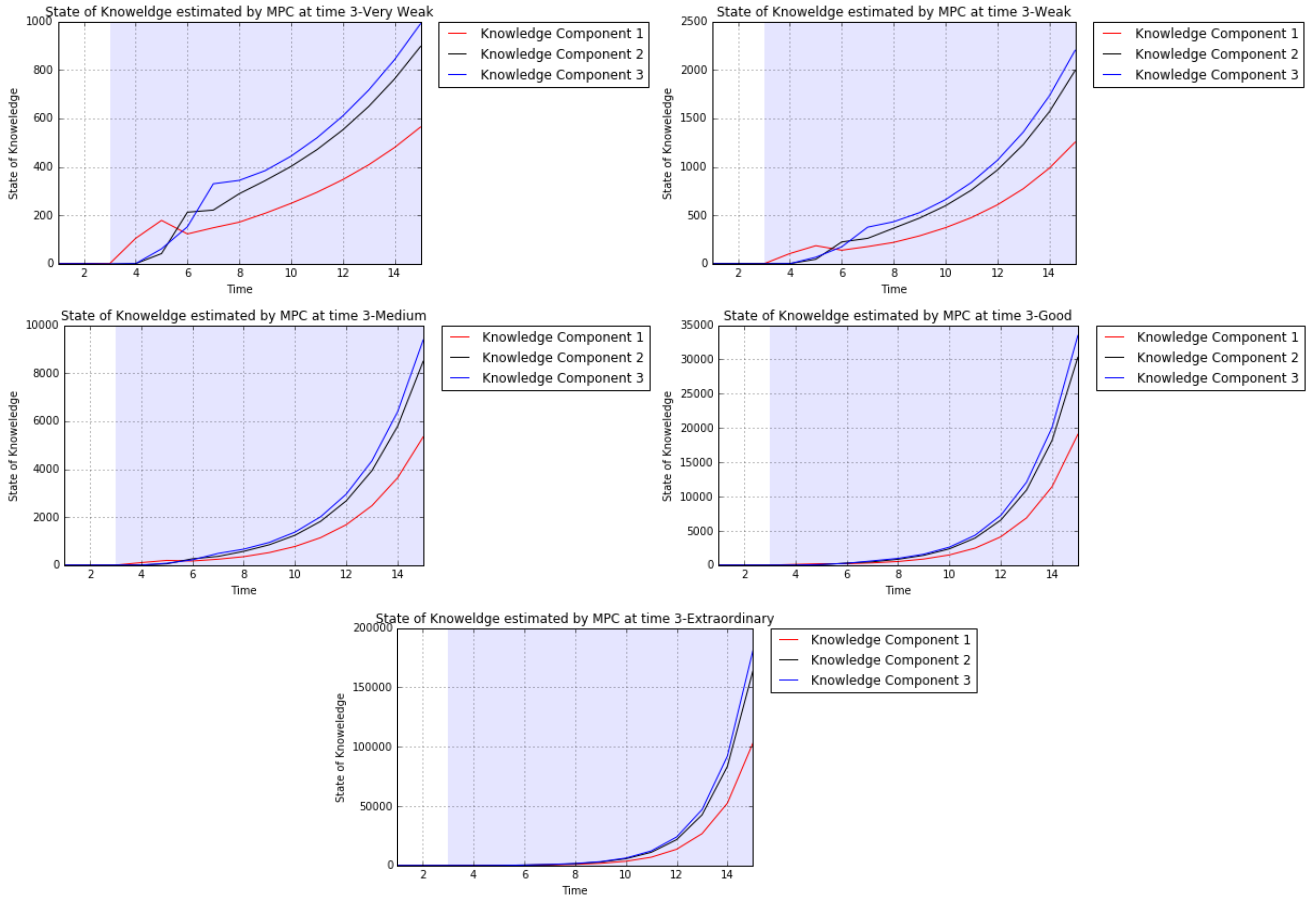

Figure D.1: Five students knowledge state with different capability in three knowledge components optimized at $T = 3$.

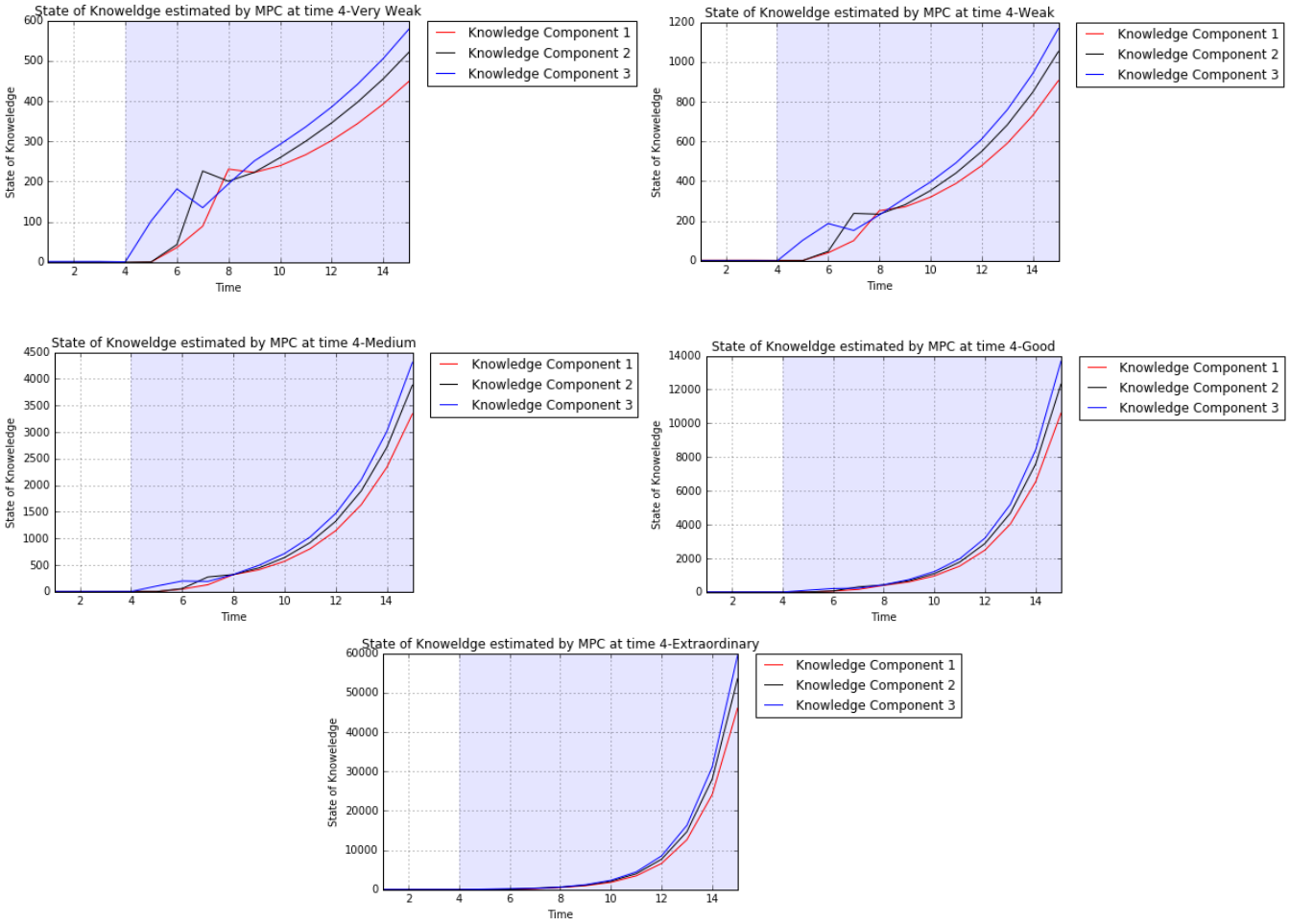

Figure D.2: Five students knowledge state with different capability in three knowledge components optimized at $T = 4$.

D.2 Partially Observable Markov Decision Process

As a sequential decision process, a POMDP is a generalization of a Markov Decision Process that information concerning states of system is incomplete (Sondik, 1978). In other words, the system states are not fully observable. For instance, students' observations such as their performance on given questions does not precisely reflect the conceptual understanding of the materials due to lucky guesses and careless mistakes. POMDPs have been utilized in wide range of practical applications in different domains(see (Lovejoy, 1991)). The main idea in partially observable systems is to exploit memory of the previous observations to reduce ambiguity of a given system (Kaelbling et al., 1998). Although this realistic extension of MDPs offers a more powerful approach to student modeling, the complexity of POMDPs dramatically increases such that finding the optimal teaching policy is computationally intractable Braziunas (2003). All the previous history of observations (i.e., students' performance) and actions (i.e., given questions) are required to obtain the optimal teaching policy.

To use POMDP approach, let $\mathcal{S}_{kjt}$ indicate the state of student j at time $t = 0, 1, \ldots, T$ in knowledge component k. In addition, $\mathcal{A}_{ijt}$ denotes question i given to student j at time t as the action and the discounted reward is denoted by $\gamma^t r(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) > 0$, where $\gamma,\ 0 < \gamma < 1$, indicates the discount factor. The main assumption is that state $\mathcal{S}_{kjt}$ behaves as a homogenous Markov chain with transition probability below:

$$\mathcal{P}_{kjt} = P(\mathcal{S}_{kjt}|\mathcal{S}_{kjt-1}, \mathcal{A}_{ijt}),$$

where $\mathcal{P}_{kjt}$ is the probability student j's knowledge transits from state $\mathcal{S}_{kjt-1}$ to $\mathcal{S}_{kjt}$ by attempting question i. Similarly, $\mathcal{Y}_{ijt}$ is the noisy measurement of performance student j on question i at time t that represents the conceptual understanding of the student denoted with $\mathcal{S}_{kjt}$. These observations, $\mathcal{Y}_{ijt}$ are binary with the following probability matrix:

$$\mathcal{O}_{ijt} = P(\mathcal{Y}_{ijt}|\mathcal{S}_{:jt}, \mathcal{A}_{ijt}),$$

Indeed, $\mathcal{O}_{ijt}$ shows the probability of the observation given the real state of students knowledge and the given question. Given the observation and the question that student j has tried, one can define a belief state $\mathcal{X}_{kjt}$ is defined as follows:

$$\mathcal{X}_{kjt} = P(\mathcal{S}_{kjt}|\mathcal{Y}_{ijt}, \mathcal{A}_{ijt}), \tag{D.1}$$

such that $\mathcal{Y}_{ijt}$ and $\mathcal{A}_{ijt}$ are observation and the action, respectively. When $\mathcal{Y}_{ijt}$ is observed,the belief state, $\mathbf{X}_{jt}$, is updated using "forward algorithm" according to:

$$\mathcal{X}_{kjt+1} = P(\mathcal{S}_{:jt+1}|\mathcal{Y}_{ijt}, \mathcal{A}_{ijt}) = \frac{\mathcal{O}_{ijt} \sum_{\mathcal{S}_{kjt}} \mathcal{P}_{kjt} \mathcal{X}_{kjt}}{\sum_{\mathcal{S'}_{kjt}} \mathcal{O}_{ijt} \sum_{\mathcal{S}_{kjt}} \mathcal{P}_{kjt} \mathcal{X}_{kjt}}, \tag{D.2}$$

To find the optimal teaching policy, an explicit t-step POMDP policy can be used by a policy tree. Given an initial belief state $\mathcal{X}_{kj0}$, the optimal t-step policy is obtained by searching through the set of all policy trees and selecting the one with the best value function. However, this method is intractable for large-scale problems. Instead of keeping all policy trees, an implicit t-step policy can be defined by using a greedy one-step ahead choice. Similar to Q-learning model, one can define a Q-value function $Q_t(\mathcal{X}_{kjt}, \mathcal{A}_{ijt})$ as a value of taking action $, \mathcal{A}_{ijt}$ at belief state $\mathcal{X}_{kjt}$ and continuing optimally for the remaining stages.

$$Q_t(\mathcal{X}_{kjt}, \mathcal{A}_{ijt}) = \sum_{\mathcal{S}_{kjt} \in \mathbf{S}} \mathcal{X}_{kjt} R(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) + \gamma \sum_{\mathcal{Y}_{ijt}} \mathcal{O}_{ijt} V^*_{t+1}(\mathcal{S}), \tag{D.3}$$

where $R(\mathcal{S}_{kjt}, \mathcal{A}_{ijt})$ is the immediate reward that student j gains in the state $\mathcal{S}_{kjt}$ by taking question $\mathcal{A}_{ijt}$ and can be defined as (7.27). In Q-value function, $V^*_{t+1}(\mathcal{S})$ is the optimal value function of the future. Algorithm (10) describes the steps of finding (sub)optimal policy using Q-value function.

Algorithm 10 Q-learning for finding optimal teaching policy

Input: $P(\mathcal{S}_{kjt}|\mathcal{S}_{kjt-1}, \mathcal{A}_{ijt})$, $P(\mathcal{Y}_{ijt}|\mathcal{S}_{:jt}, \mathcal{A}_{ijt})$ and $R(\mathcal{S}_{kjt}, \mathcal{A}_{ijt})$.
Output: Optimal Policy $\pi^*_Q(\mathcal{S})$.
POMDP()

1. $t \leftarrow 0$
2. Initialize $V_t(\mathcal{S}) = 0 \; \forall \; \mathcal{S}_{kjt} \in \mathbf{S}$, $\mathbf{A}^+ = \mathbf{A}$ and $\mathcal{X}_{kj0}$
3. **do**
4. $t \leftarrow t + 1$
5. **for** all belief states $\mathcal{X}_{kjt} \in \mathbf{X}$
6. **for** all actions $\mathcal{A}_{ijt} \in \mathbf{A}^+$
7. $Q_\pi(\mathcal{X}_{kjt}, \mathcal{A}_{ijt}) = \sum_{\mathcal{S}_{kjt} \in \mathbf{S}} \mathcal{X}_{kjt} R(\mathcal{S}_{kjt}, \mathcal{A}_{ijt}) + \gamma \sum_{\mathcal{Y}_{ijt}} \mathcal{O}_{ijt} V^*_{t+1}(\mathcal{S})$
8. $V_t(\mathcal{S}) = \max_{\mathcal{A}_{ijt} \in \mathcal{A}} Q(\mathcal{X}_{kjt+1}, \mathcal{A}_{ijt+1})$
9. $\pi_Q(\mathcal{S}) = \arg\max_{\mathcal{A}_{ijt} \in \mathcal{A}} Q(\mathcal{X}_{kjt+1}, \mathcal{A}_{ijt+1})$
10. $\mathcal{X}_{kjt+1} = P(\mathcal{S}_{:jt+1}|\mathcal{Y}_{ijt}, \mathcal{A}_{ijt}) = \frac{\mathcal{O}_{ijt} \sum_{\mathcal{S}_{kjt}} \mathcal{P}_{kjt} \mathcal{X}_{kjt}}{\sum_{\mathcal{S'}_{kjt}} \mathcal{O}_{ijt} \sum_{\mathcal{S}_{kjt}} \mathcal{P}_{kjt} \mathcal{X}_{kjt}}$
11. $\mathbf{A}^+ \leftarrow \mathbf{A}^+ - \{\mathcal{A}^*_{ijt}\}$
12. **while** $\|V_t(\mathcal{S}) - V_{t-1}(\mathcal{S})\| \leq \epsilon$
13. return $\pi^*_Q(\mathcal{S})$

Bibliography

Bajraktarevic[1], N., W. Hall[1], and P. Fullick (2003). Incorporating learning styles in hypermedia environment: Empirical evaluation. 7.1

Beal, C. R., P. R. Cohen, et al. (2012). Teach ourselves: Technology to support problem posing in the stem classroom. *Creative Education 3*(04), 513. 2.3, 2.5

Bekele, R. and W. Menzel (2005). A bayesian approach to predict performance of a student (bapps): A case with ethiopian students. *algorithms 22*(23), 24. 6.1

Bell, R. M. and Y. Koren (2007). Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 43–52. IEEE. 6.2

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning 3*(1), 1–122. 6.4.5

Bradley, J. K., A. Kyrola, D. Bickson, and C. Guestrin (2011). Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379*. 5.2

Braha, D. and O. Maimon (1997). The design process: properties, paradigms, and structure. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 27*(2), 146–166. 2.3

Bransford, J. D., A. L. Brown, and R. R. Cocking (1999). *How people learn: Brain, mind, experience, and school*. National Academy Press. 2.3

Braziunas, D. (2003). Pomdp solution methods. *University of Toronto, Tech. Rep*. 7.2, D.2

Brindley, C. and S. Scoffield (1998). Peer assessment in undergraduate programmes. *Teaching in higher education 3*(1), 79–90. 2.3

Brinton, C. G. and M. Chiang (2015). Mooc performance prediction via clickstream data and social learning networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2299–2307. IEEE. 6.2

Brunskill, E. and S. Russell (2010). Partially observable sequential decision making for problem selection in an intelligent tutoring system. In *Educational Data Mining 2011*. 3.4

Cassidy, S. (2006). Developing employability skills: Peer assessment in higher education. *Education+ Training 48*(7), 508–517. 2.3

Cen, H., K. Koedinger, and B. Junker (2006). Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pp. 164–175. Springer. 6.1, 6.2

Cetintas, S., L. Si, Y. P. P. Xin, and C. Hord (2010). Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques. *IEEE Transactions on Learning Technologies 3*(3), 228–236. 6.1

Chi, M., K. VanLehn, D. Litman, and P. Jordan (2011). An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach. *International Journal of Artificial Intelligence in Education 21*(1-2), 83–113. 7.1, 7.2

Cichocki, A. and P. Anh-Huy (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences 92*(3), 708–721. 5.2, 6.4.4

Clement, B., D. Roy, P.-Y. Oudeyer, and M. Lopes (2013). Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*. 2.6, 3.4

Cohen, G. L. and J. Garcia (2008). Identity, belonging, and achievement a model, interventions, implications. *Current Directions in Psychological Science 17*(6), 365–369. 3.1

Comon, P. (2014). Tensors: a brief introduction. *IEEE Signal Processing Magazine 31*(3), 44–53. 6.3

Corbett, A. T. and J. R. Anderson (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction 4*(4), 253–278. 3.2

Cotter, A., O. Shamir, N. Srebro, and K. Sridharan (2011). Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pp. 1647–1655. 5.2

d Baker, R. S., A. T. Corbett, and V. Aleven (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*, pp. 406–415. Springer. 3.2, 6.2

Daubigney, L., M. Geist, and O. Pietquin (2013). Model-free pomdp optimisation of tutoring systems with echo-state networks. In *Proc. 14th SIGDial Meeting on Discourse and Dialogue*, pp. 102–106. 3.4

Davenport, M. and J. Romberg. An overview of low-rank matrix recovery from incomplete observations. 1.2.2, 3.3

DILLON, J. (1988). 6. questioning in education. *Questions and questioning*, 98. 1.2.2

diSessa, A. A. (2002). Students' criteria for representational adequacy. *Symbolizing, modeling and tool use in mathematics education*, 105–129. 2.3

Dorça, F. A., L. V. Lima, M. A. Fernandes, and C. R. Lopes (2013). Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis. *Expert Systems with Applications 40*(6), 2092–2101. 7.1, 7.2

Duschl, R. A., H. A. Schweingruber, A. W. Shouse, et al. (2007). *Taking science to school: Learning and teaching science in grades K-8*. National Academies Press. 1.3.1, 2.3

Edmunds, B., Z. Peng, and W. Yin (2016). Tmac: A toolbox of modern async-parallel, coordinate, splitting, and stochastic methods. *arXiv preprint arXiv:1606.04551*. 5.2

Elliot, A. J. and C. S. Dweck (2013). *Handbook of competence and motivation*. Guilford Publications. 3.1

English, L. D. (1998). Children's problem posing within formal and informal contexts. *Journal for Research in mathematics Education*, 83–106. 2.3

Evgeniou, A. and M. Pontil (2007). Multi-task feature learning. *Advances in neural information processing systems 19*, 41. 6.3

Feng, M., N. Heffernan, and K. Koedinger (2009). Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction 19*(3), 243–266. 4.1, 6.1

Fossati, D., B. Di Eugenio, S. Ohlsson, C. Brown, and L. Chen (2015). Data driven automatic feedback generation in the ilist intelligent tutoring system. *Technology, Instruction, Cognition and Learning 10*(1), 5–26. 7.1, 7.2

Frolov, E. and I. Oseledets (2016). Tensor methods and recommender systems. *arXiv preprint arXiv:1603.06038*. 6.3, 6.3.2, 6.3.2

Ghahramani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence 15*(01), 9–42. 4.2.1, B.1

González-Brenes, J., Y. Huang, and P. Brusilovsky (2014). General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *The 7th International Conference on Educational Data Mining*, pp. 84–91. University of Pittsburgh. 3.2, 6.2

Gorski, J., F. Pfeuffer, and K. Klamroth (2007). Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research 66*(3), 373–407. 6.4, 6.4.1

Graesser, A. C., M. W. Conley, and A. Olney (2012). Intelligent tutoring systems. 1.2, 5.1

Graf, S. et al. (2012). Personalized learning systems. In *Encyclopedia of the Sciences of Learning*, pp. 2594–2596. Springer. 1.2, 5.1

Graf, S., T.-C. Liu, and K. Kinshuk (2008). Interactions between students learning styles, achievement and behaviour in mismatched courses. In *Proceedings of the international conference on cognition and exploratory learning in digital age (CELDA 2008). IADIS International Conference*, pp. 223–230. Citeseer. 7.1

Graf, S. and S. Viola (2009). Automatic student modelling for detecting learning style preferences in learning management systems. In *Proc. international conference on cognition and exploratory learning in digital age*, pp. 172–179. 7.1

Grasedyck, L., D. Kressner, and C. Tobler (2013). A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen 36*(1), 53–78. 6.3

Haider, M., A. Sinha, and B. Chaudhary (2010). An investigation of relationship between learning styles and performance of learners. *International Journal of Engineering Science and Technology 2*(7), 2813–2819. 7.1

Hsieh, C.-J., K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan (2008). A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pp. 408–415. ACM. 5.2, 6.4.4

Hsieh, C.-J. and I. S. Dhillon (2011). Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1064–1072. ACM. 5.2

Joachims, T. and C. EDU (2015). Learning representations of student knowledge and educational content. In *International Conference on Machine Learning WorkshopMachine Learning for Education*. 3.1

Kaelbling, L. P., M. L. Littman, and A. R. Cassandra (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence 101*(1), 99–134. 7.2, D.2

Kar, T., E. Özdemir, A. S. İpek, and M. Albayrak (2010). The relation between the problem posing and problem solving skills of prospective elementary mathematics teachers. *Procedia-Social and Behavioral Sciences 2*(2), 1577–1583. 2.3

Khajah, M., R. M. Wing, R. V. Lindsey, and M. C. Mozer (2014). Incorporating latent factors into knowledge tracing to predict individual differences in learning. In *Proceedings of the 7th International Conference on Educational Data Mining*. Citeseer. 3.4

Khajah, M. M., Y. Huang, J. P. González-Brenes, M. C. Mozer, and P. Brusilovsky (2014). Integrating knowledge tracing and item response theory: A tale of two frameworks. *Personalization Approaches in Learning Environments*, 7. 3.1, 3.3, 3.4, 5.1, 5.2, 6.2

Koedinger, K. R., R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper (2010). A data repository for the edm community: The pslc datashop. *Handbook of educational data mining 43*. 4.1

Koedinger, K. R., J. Kim, J. Z. Jia, E. A. McLaughlin, and N. L. Bier (2015). Learning is not a spectator sport: Doing is better than watching for learning from a mooc. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pp. 111–120. ACM. 2.1

Komarek, P. and A. W. Moore (2005). Making logistic regression a core data mining tool with tr-irls. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 4–pp. IEEE. 6.2

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD) 4*(1), 1. 6.2

Koren, Y., R. Bell, C. Volinsky, et al. (2009). Matrix factorization techniques for recommender systems. *Computer 42*(8), 30–37. 6.2

Kotsiantis, S. B. and P. E. Pintelas (2005). Predicting students marks in hellenic open university. In *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, pp. 664–668. IEEE. 6.1

Kurucz, M., A. A. Benczúr, and K. Csalogány (2007). Methods for large scale svd with missing values. In *Proceedings of KDD cup and workshop*, Volume 12, pp. 31–38. Citeseer. 6.3

Lan, A. S., C. Studer, and R. G. Baraniuk (2014a). Quantized matrix completion for personalized learning. *arXiv preprint arXiv:1412.5968*. 3.3, 4.1, 5.2, 6.2

Lan, A. S., C. Studer, and R. G. Baraniuk (2014b). Time-varying learning and content analytics via sparse factor analysis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 452–461. ACM. 1.1, 2.6, 3.3, 3.4, 3.5, 6.2, 6.3.2

Lan, A. S., C. Studer, A. E. Waters, and R. G. Baraniuk (2014). Tag-aware ordinal sparse factor analysis for learning and content analytics. *arXiv preprint arXiv:1412.5967*. 5.2

Lan, A. S., A. E. Waters, C. Studer, and R. G. Baraniuk (2013). Sparse factor analysis for learning and content analytics. *arXiv preprint arXiv:1303.5685*. 3.2, 6.3.2

Lan, A. S., A. E. Waters, C. Studer, and R. G. Baraniuk (2014). Sparse factor analysis for learning and content analytics. *The Journal of Machine Learning Research 15*(1), 1959–2008. 1.1, 1.3.2, 2.6, 3.1, 3.3, 3.5, 5.1, 5.2, 5.2, 6.1, 6.2, 6.3.1, 6.3.2, 6.6, 8.2.3

Lavy, I. and I. Bershadsky (2003). Problem posing via "what if not?" strategy in solid geometry-case study. *The Journal of Mathematical Behavior 22*(4), 369–387. 2.3

Lavy, I. and A. Shriki (2010). Engaging in problem posing activities in a dynamic geometry setting and the development of prospective teachers' mathematical knowledge. *The Journal of Mathematical Behavior 29*(1), 11–24. 2.3

Li, M.-C. and C.-C. Tsai (2013). Game-based learning in science education: A review of relevant research. *Journal of Science Education and Technology 22*(6), 877–898. 2.3

Li, N., W. Cohen, K. R. Koedinger, and N. Matsuda (2010). A machine learning approach for automatic student model discovery. In *Educational Data Mining 2011*. 5.2

Li, N., W. W. Cohen, K. R. Koedinger, N. Matsuda, and C. Mellon (2011). A machine learning approach for automatic student model discovery. In *EDM*, pp. 31–40. 5.1

Lindsey, R. V., M. Khajah, and M. C. Mozer (2014). Automatic discovery of cognitive skills to improve the prediction of student learning. In *Advances in neural information processing systems*, pp. 1386–1394. 4.1

Linn, M. C., D. Clark, and J. D. Slotta (2003). Wise design for knowledge integration. *Science education 87*(4), 517–538. 2.3

Linnenbrink, E. A. and P. R. Pintrich (2004). Role of affect in cognitive processing in academic contexts. *Motivation, emotion, and cognition: Integrative perspectives on intellectual functioning and development*, 57–87. 3.1

Liu, J., P. Musialski, P. Wonka, and J. Ye (2013). Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*(1), 208–220. (document), 3.2, 6.3, 6.4

Liu, T.-C., S. Graf, et al. (2009). Coping with mismatched courses: students behaviour and performance in courses mismatched to their learning styles. *Educational Technology Research and Development 57*(6), 739–752. 7.1

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research 28*(1), 47–65. 7.2, D.2

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. Scokaert (2000). Constrained model predictive control: Stability and optimality. *Automatica 36*(6), 789–814. 7.5

Mestre, J. P. (2002). Probing adults' conceptual understanding and transfer of learning via problem posing. *Journal of Applied Developmental Psychology 23*(1), 9–50. 2.3

Minaei-Bidgoli, B., D. A. Kashy, G. Kortemeyer, and W. F. Punch (2003). Predicting student performance: an application of data mining methods with an educational web-based system. In *Frontiers in education, 2003. FIE 2003 33rd annual*, Volume 1, pp. T2A–13. IEEE. 6.1

Mishra, S. and S. Iyer (2015a). An exploration of problem posing-based activities as an assessment tool and as an instructional strategy. *Research and Practice in Technology Enhanced Learning 10*(1), 1. 2.3

Mishra, S. and S. Iyer (2015b). Question-posing strategies used by students for exploring data structures. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 171–176. ACM. 1.3.1

Mitros, P. (2015). Learnersourcing of complex assessments. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pp. 317–320. ACM. 2.5

Necoara, I. and D. Clipici (2013). Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed mpc. *Journal of Process Control 23*(3), 243–253. 5.2

Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization 22*(2), 341–362. 7.3.2

Ning, R., A. E. Waters, C. Studer, and R. G. Baraniuk (2015). Sprite: A response model for multiple choice testing. *arXiv preprint arXiv:1501.02844*. 3.4

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, Volume 2007, pp. 5–8. 6.2

Pavlik Jr, P. I., H. Cen, and K. R. Koedinger (2009). Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*. 3.3, 6.2

Piech, C., J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein (2015). Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, pp. 505–513. 3.1, 3.3, 3.5, 3.5.4, 4.1, 6.2

Polyak, B. T. (1987). Introduction to optimization. translation series in mathematics and engineering. *Optimization Software*. 7.3.2

Profetto-McGrath, J., K. B. Smith, R. A. Day, and O. Yonge (2004). The questioning skills of tutors and students in a context based baccalaureate nursing program. *Nurse Education Today 24*(5), 363–372. 2.3

Purchase, H. C. (2000). Learning about interface design through peer assessment. *Assessment & Evaluation in Higher Education 25*(4), 341–352. 2.3

Rafailidis, D. and P. Daras (2013). The tfc model: Tensor factorization and tag clustering for item recommendation in social tagging systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems 43*(3), 673–688. 6.3

Rafferty, A. N., E. Brunskill, T. L. Griffiths, and P. Shafto (2011). Faster teaching by pomdp planning. In *Artificial intelligence in education*, pp. 280–287. Springer. 2.6, 3.4, 5.1, 7.2

Rafferty, A. N., M. M. LaMar, and T. L. Griffiths (2015). Inferring learners' knowledge from their actions. *Cognitive science 39*(3), 584–618. 3.2

Recht, B., C. Re, S. Wright, and F. Niu (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 693–701. 5.2

Reckase, M. (2009). *Multidimensional item response theory*, Volume 150. Springer. 6.2

Redfield, D. L. and E. W. Rousseau (1981). A meta-analysis of experimental research on teacher questioning behavior. *Review of educational research 51*(2), 237–245. 1.2.2

Richtárik, P. and M. Takáč (2012). Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 1–52. 5.2

Richtárik, P. and M. Takáč (2016). Parallel coordinate descent methods for big data optimization. *Mathematical Programming 156*(1-2), 433–484. 5.2

Rothstein, D. and L. Santana (2011). *Make just one change: Teach students to ask their own questions.* ERIC. 1.2.2

Ruiz Palmero, J. and J. Sanches Rodriguez (2012). Peer assessment in higher education. a case study. *New Educational Review 27*(1), 247–255. 2.3

Sahebi, S., Y. Huang, and P. Brusilovsky (2014). Predicting student performance in solving parameterized exercises. In *Intelligent Tutoring Systems*, pp. 496–503. Springer. 5.1

Sallinen, S., N. Satish, M. Smelyanskiy, S. S. Sury, and C. Ré. High performance parallel stochastic gradient descent in shared memory. 5.2

Schraagen, J. M., S. F. Chipman, and V. L. Shalin (2000). *Cognitive task analysis.* Psychology Press. 3.5

Shen, S. and M. Chi (2016). Reinforcement learning: the sooner the better, or the later the better? In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pp. 37–44. ACM. 7.2

Shute, V. J. and R. Torres (2012). Where streams converge: Using evidence-centered design to assess quest to learn. *Technology-based assessments for 21st century skills: Theoretical and practical implications from modern research*, 91–124. 2.3

Silver, E. A. and J. Cai (1996). An analysis of arithmetic problem posing by middle school students. *Journal for Research in Mathematics Education*, 521–539. 1.3.1

Silver, E. A., J. Mamona-Downs, S. S. Leung, and P. A. Kenney (1996). Posing mathematical problems: An exploratory study. *Journal for research in mathematics Education*, 293–309. 2.3

Sondik, E. J. (1978). The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research 26*(2), 282–304. 7.2, D.2

Soni, A., S. Jain, J. Haupt, and S. Gonella (2016). Noisy matrix completion under sparse factor models. *IEEE Transactions on Information Theory 62*(6), 3636–3661. 3.3, 6.2

Take, W. W. I. (2012). 21st century skills: The challenges ahead. *Challenge 12*, 1. 2.3

Thai-Nghe, N. (2011). Predicting student performance in an intelligent tutoring system. *Department of Computer Science, University of Hildesheim, Hildesheim, Germany*. 1.1, 3.5, 4.1, 6.1, 6.2

Thai-Nghe, N., A. Busche, and L. Schmidt-Thieme (2009). Improving academic performance prediction by dealing with class imbalance. In *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pp. 878–883. IEEE. 6.1

Thai-Nghe, N., L. Drumond, T. Horváth, A. Krohn-Grimberghe, A. Nanopoulos, and L. Schmidt-Thieme (2011). Factorization techniques for predicting student performance. *Educational Recommender Systems and Technologies: Practices and Challenges*, 129–153. 6.1

Thai-Nghe, N., L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme (2010). Recommender system for predicting student performance. *Procedia Computer Science 1*(2), 2811–2819. 1.1, 3.5, 6.1, 6.2

Topping, K. J. and S. W. Ehly (2001). Peer assisted learning: A framework for consultation. *Journal of Educational and Psychological Consultation 12*(2), 113–132. 2.3

Toscher, A. and M. Jahrer (2010). Collaborative filtering applied to educational data mining. *KDD cup*. 6.1

van De Sande, B. (2013). Properties of the bayesian knowledge tracing model. *JEDM-Journal of Educational Data Mining 5*(2), 1–10. 2.6

Van Der Meij, H. (1994). Student questioning: A componential analysis. *Learning and individual Differences 6*(2), 137–161. 1.2.2

Van Lehn, K., M. Chi, W. Baggett, and R. Murray (1995). Progress report: Towards a theory of learning during tutoring. *Pittsburgh, PA: Learning Research and Development Center, University of Pittsburgh*. 2.3

Van Zundert, M., D. Sluijsmans, and J. Van Merriënboer (2010). Effective peer assessment processes: Research findings and future directions. *Learning and Instruction 20*(4), 270–279. 2.3

Veenman, M. V. (2012). Metacognition in science education: Defi nitions, constituents, and their intricate relation with cognition. In *Metacognition in science education*, pp. 21–36. Springer. 2.3

Waters, A. E., D. Tinapple, and R. G. Baraniuk (2015). Bayesrank: A bayesian approach to ranked peer grading. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pp. 177–183. ACM. 4.1

White, B. Y. and J. R. Frederiksen (1998). Inquiry, modeling, and metacognition: Making science accessible to all students. *Cognition and instruction 16*(1), 3–118. 2.3

Williams, J. J., J. Kim, A. Rafferty, S. Maldonado, K. Z. Gajos, W. S. Lasecki, and N. Heffernan (2016). Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pp. 379–388. ACM. 2.2

Williams, J. J., M. Krause, P. Paritosh, J. Whitehill, J. Reich, J. Kim, P. Mitros, N. Heffernan, and B. C. Keegan (2015). Connecting collaborative & crowd work with online education. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, pp. 313–318. ACM. 2.6, 8.2.1

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming 151*(1), 3–34. 5.2, 5.3, 6.4.4, 7.3.2

Yu, H.-F., C.-J. Hsieh, I. Dhillon, et al. (2012). Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pp. 765–774. IEEE. 5.2

Yu, H.-F., H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, et al. (2010). Feature engineering and classifier ensemble for kdd cup 2010. In *Proceedings of the KDD Cup 2010 Workshop*, pp. 1–16. 6.1

Yudelson, M. V., K. R. Koedinger, and G. J. Gordon (2013). Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education*, pp. 171–180. Springer. 2.6, 3.2, 6.2

Zhuang, Y., W.-S. Chin, Y.-C. Juan, and C.-J. Lin (2013). A fast parallel sgd for matrix factorization in shared memory systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 249–256. ACM. 5.2